2011

Surface parameterization over regular domains

Shenghua Wan
Louisiana State University and Agricultural and Mechanical College, swan2@lsu.edu

SURFACE PARAMETERIZATION OVER REGULAR DOMAINS

A Thesis
Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Master of Science in Electrical Engineering
in
The Department of Electrical & Computer Engineering

by

Shenghua Wan
Bachelor of Computer Science, Harbin Institute of Technology, 2009
December, 2011

Dedicated to my family and friends

Acknowledgements

At LSU, I have been very fortunate in being surrounded by many supportive friends and colleagues. This thesis would not have been made possible without their kind advice and help.

I would like to express my sincere appreciation to my advisor Dr. Xin Li for the valuable academic suggestions and persistent patience guiding me throughout the research and preparation for this thesis. His technical advice and expertise in this field profoundly influenced me and this work recorded herein. Without his encouragement and inspiration, this thesis would not have been completed.

Meanwhile, I would like give thanks to Dr. Hongchao Zhang for the discussion on the optimization part and providing us an excellent derivative-free optimization solver, which makes the whole framework in this thesis possible. Thanks to Dr. Jagannathan Ramanujam and Dr. Hsiao-Chun Wu for the constructive instructions on this thesis.

Thanks also go to my cherished family who always trust and support me through the years, especially my beloved wife. I would not be able to focus on the work and chase my dream without their sustainment.

I would also like to express my gratitude to all members in Geometric and Visual Computing Group in LSU. Many thanks are given to Zhao Yin, Li Wei and Kang Zhang for the collaboration for polycube domain construction and optimization and common domain for multiple objects. I really appreciate Wuyi Yu and Huanhuan Xu for the generous willingness and patience to share the ideas and discuss problems in research and coursework.

Thank you to my friends, Qing Huang, Dongsheng Guan, Chuan Cai and others. Thanks for helping me accommodate to the life at LSU.

Table of Contents

List of Tables

List of Figures

Abstract

Surface parameterization has been widely studied and it has been playing a critical role in many geometric processing tasks in graphics, computer-aided design, visualization, vision, physical simulation and etc. Regular domains, such as polycubes, are favored due to their structural regularity and geometric simplicity. This thesis focuses on studying the surface parameterization over regular domains, i.e. polycubes, and develops effective computation algorithms. Firstly, the motivation for surface parameterization and polycube mapping is introduced. Secondly, we briefly review existing surface parameterization techniques, especially for extensively studied parameterization algorithms for topological disk surfaces and parameterizations over regular domains for closed surfaces. Then we propose a polycube parameterization algorithm for closed surfaces with general topology. We develop an efficient optimization framework to minimize the angle and area distortion of the mapping. Its applications on surface meshing, inter-shape morphing and volumetric polycube mapping are also discussed.

Chapter 1 Introduction

1.1 Motivations and Thesis Scope

A parameterization of a surface can be viewed as a one-to-one mapping from the surface to a specific domain. In general, the parametric domain itself is a surface and therefore constructing a parameterization means mapping one surface to another. Usually, the surfaces are either represented or approximated by triangular meshes, hence the mappings are piecewise linear. Computing the parameterization of 3D shapes (surfaces/solids) on specific domains is an important problem in shape modeling, which can facilitate many computer graphics and geometric processing tasks, including texture mapping, data fitting, re-parameterization of spline surfaces, physical simulation, and repair of CAD models.

Regular domains, including polycubes, are among the most favorable parametric domains by researchers due to their simplicity and regularity, which reduces the complexity to build models and to do simulations. However, parameterizations almost always introduce angle or area distortion. And a good mapping favored by applications is the one which minimizes these distortions to some extent.

Polycube is the surface of a solid consist of a few solid cubes (see Figure 1.1). Polycube mapping was first introduced by [**1**]. It parameterizes a closed surface onto a polycube domain. A polycube has the same topology of the given surface, and it is usually constructed to approximate the geometry of the surface. Therefore, the surface parameterization on a polycube domain often has much smaller distortion than that on a planar domain. Meanwhile, the polycube domain still possesses great regularity; each sub-patch is a rectangle; transitions between adjacent patches are simple rotation and translation except on corner points. Due to these advantages, the polycube mapping has been used in many graphics and shape modeling

applications such as texture mapping [**1**] and synthesis [**2**], shape morphing [**3**], spline construction [**4**] [**5**], and volumetric matching [**6**] [**7**].

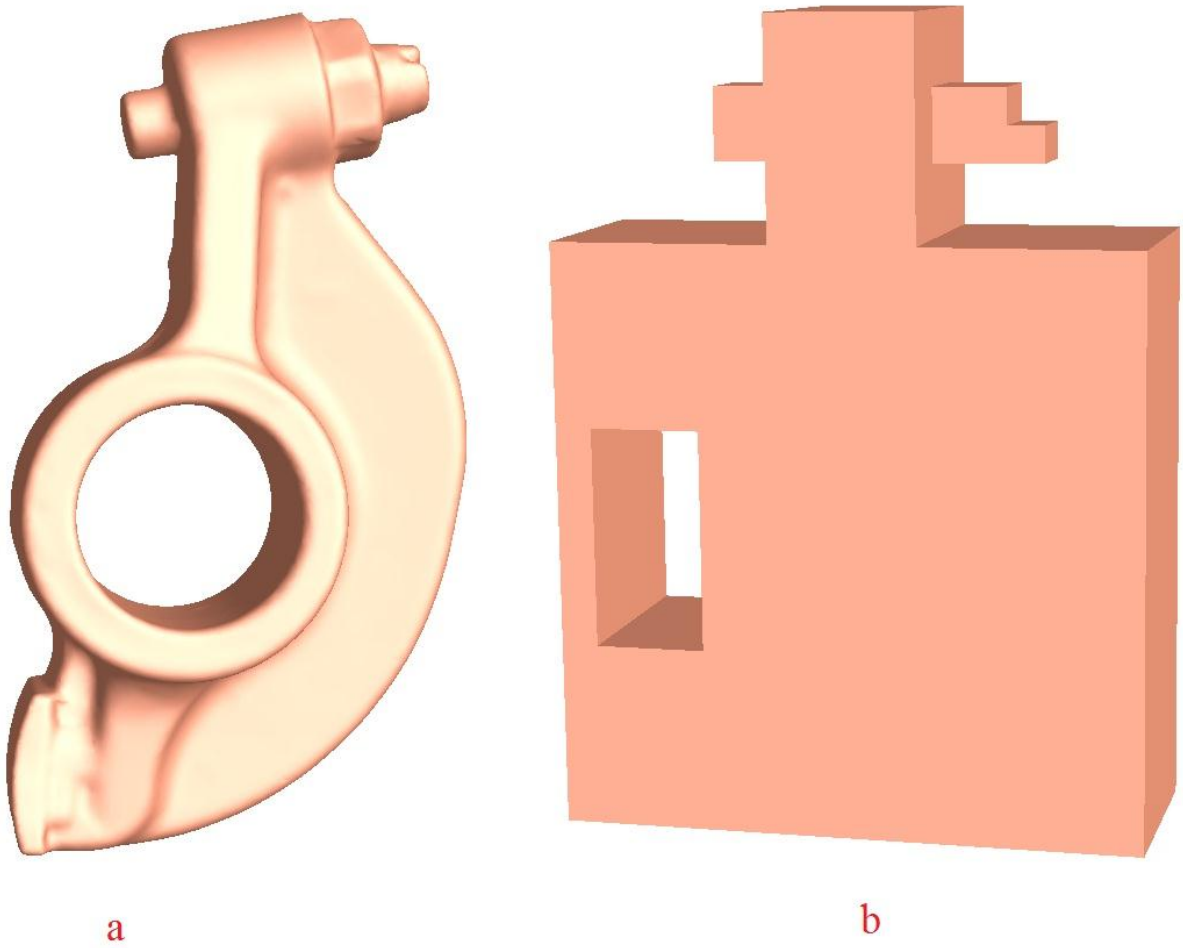

Figure 1.1. A model and its polycube.
a) is Rockarm, whose genus=1. (b) is the polycube for it with the same genus.

Intuitively, on one hand, the more cubes one uses to construct the polycube, the better the domain can approximate the original model, which brings the parameterization very small area and angle distortion. However, corner points are singularity points of the parameterization. They are undesirable in many tasks such as spline construction [**4**] [**5**], physics-based simulations [**8**], etc. On the other hand, if one uses fewer cubes to construct a simpler domain with fewer corner points, the parameterization will possess larger distortion due to the dissimilarity of geometric structures between the model and the domain shape. Therefore, when a fundamental question is asked: What is the optimal polycube domain? A

reasonable answer can be an optimized balance between the singularity number and mapping distortion. More specifically, we try to solve the following problem: given a surface S and a budget n of the singularity point number, what is the optimal shape of the polycube domain P so that the parameterization $f: S \to P$ has the least distortion and P has no more than n corners?

Now the optimal polycube maps can be formulated as solving $argmin\, E(P, f)$ for a given shape S, where energy function E is defined on any mapping $f: S \to P$ and P is a polycube with n corners. Since the domain P is part of the optimization, it is extremely difficult. We restrict our optimization to a subspace of this problem, which we call a *topology-preserving polycube mapping*. Specifically, given an initial polycube domain $P = \{P_i\}$, the *topology* of the polycube P is defined by its dual graph (see Figure 1.2) $DM = \{DV, DE\}$. $DV = \{dv_1, \dots, dv_n\}$ contains nodes corresponding to rectangle sub-patches $\{P_i\}$. DE is a set of edges: an edge $[dv_i, dv_j] \in DE$, if P_i and P_j are adjacent to each other. We say two polycubes $P = \{P_1, \dots, P_n\}$ and $Q = \{Q_1, \dots, Q_m\}$ are *topologically equivalent*, if their dual graphs DP and DQ are isomorphic.

Therefore, given an initial polycube P, our goal is to find the optimal polycube P' and the mapping f that minimizes distortion $E(P, f)$, in the same topological equivalence class (without changing the structure of its dual graph).

The quality of surface parameterization is measured by distortions, including angle distortion and area distortion. To some extent, the distortion reveals the stretching of the parameterization. How to obtain a low-distortion parameterization becomes the focus of the past and recent researchers. Therefore, efficient and robust algorithms for establishing low-distortion parameterization are demanded. In this thesis, the focus will be the establishment of low-distortion surface parameterizations over polycube domains.

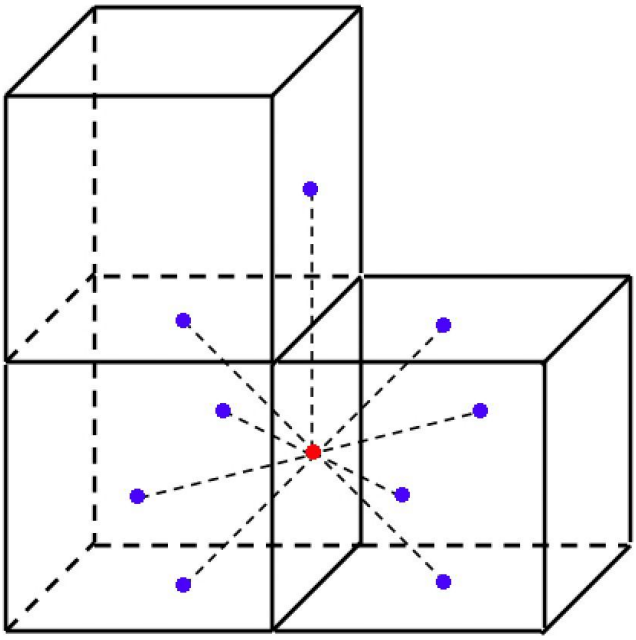

Figure 1.2. Part of the dual graph
It is corresponding to one facet (red node) and its neighboring facets(blue nodes).

1.2 Related Work

1.2.1 Surface Parameterization

Theories and technologies in surface parameterization have been widely studied and they have been playing a critical role in many geometric processing tasks in graphics, CAGD, visualization, vision, medical imaging, physical simulation, and etc. Many effective techniques have been developed to solve the parameterization under different distortion metrics with different boundary conditions. A thorough review is beyond the scope of this paper, and we refer to three great surveys/tutorials of surface mapping and their applications in [**9**] [**10**], and [**11**]. One widely used scalar function used for constructing low-distorted surface parameterization is the harmonic function. The discrete harmonic map was first proposed by Pinkall and Polthier [**12**] and introduced to the computer graphics field by Eck et al. [**13**]. By discretizing the energy defined in [**12**], Desbrun et al. [**14**] constructed free

boundary harmonic maps. Harmonic maps are preferable due to at least two important reasons: (1) it is meaningful from physics' point of view. A harmonic map minimizes the Dirichlet energy and leads to a minimal surface [**12**]; (2) it can be easily discretized and efficiently calculated from the computational aspect. A discrete harmonic map can be approximated either through FEM analysis of the harmonic energy [**13**], or via mimicking the mean value property of harmonic functions [**15**]. The computation of discrete harmonic mapping can be written as the optimization of a quadratic energy and be efficiently solved as a sparse linear system.

1.2.2 Polycube Mapping

As a useful parametric domain, polycube maps have been studied in many different shape modeling applications. Tarini et al. invented the concept of polycube map and applied it to the texture mapping and synthesis [**1**]. Fan et al. extended it to generate cross parameterization and morphing by mapping surfaces to polycubes then composing the map by finding the correspondence between them [**3**]. In these approaches, polycube maps are computed by extrinsic methods such as projections. Wang et al. introduced an intrinsic method for polycube maps and built splines representation on the polycube parametric domain [**4**]. Compared with extrinsic methods, the intrinsic approach reduced the mapping distortion significantly.

Later, Wang et al. developed user controllable polycube maps for manifold spline construction [**5**]. Both approaches required much user involvement in polycube design. Lin et al. presented an automatic polycube mapping approach, but the bijectivity was not guaranteed [**16**]. After that, He et al. presented a divide-and-conquer approach for automatic polycube map construction [**17**]. In that paper, the bijectivity was guaranteed and the mapping had shown low angle and area distortion. Then, Han et al. applied volumetric polycube maps to

construct hexahedral shell mesh [**18**]. Recently, Xia et al. introduced an editable polycube mapping, based on a divide-and-conquer strategy, which gave much more control over the quality of the induced subdivision surface and made processing of large models with complex geometry and topology feasible [**19**].

1.3 Thesis Contribution

In this thesis, surface parameterization over one regular domain is introduced, i.e. polycube mapping. Considering the singularity and the distortion simultaneously, we propose an iterative framework to compute the optimal polycube mapping, whose corner points, or singularity points, is constrained by the given budget. Based on harmonic mapping, a topology preserving optimization algorithm is employed to lower the combined angle and area distortion, with the help from efficient polycube mapping updating and a derivative-free solver [**20**]. This thesis proposes the problem to obtain optimal polycube mapping and reveals the way to obtain the solution. The experiment results indicate these parameterization and optimization approaches are effective and efficient.

1.4 Organization

There are four chapters in this thesis. Chapter 1 is a brief introduction of surface parameterization over regular domain and its applications. Chapter 2 mainly gives a review of some approaches of surface parameterization and the measurement of distortion metrics. Chapter 3 presents the algorithm to establish polycube mapping and optimization preserving the topology of given shapes. Chapter 4 is the summary of the whole thesis.

Chapter 2 Surface Parameterization and Distortions

Surface parameterization finds a better way to represent the geometric model for applications. However, distortions always exist in the surface parameterization. In this chapter, the thesis is going to introduce the parameterization of surface patches on planar domain, and the parameterization of closed surface over regular domain. Then it will review the measurement of the distortions, which indicates the quality of the parameterization. It will focus on reviewing the theories and the concepts.

2.1 General Goals

Establishing a parameterization for given shapes means attaching a coordinate system to it. There are many possible applications based on such a coordinate system. One of the main applications of surface parameterization is texture mapping. Figure 2.1 shows an example of a parameterization implemented using harmonic mapping, through which the female image pixels are attached to the 3D model Venus.

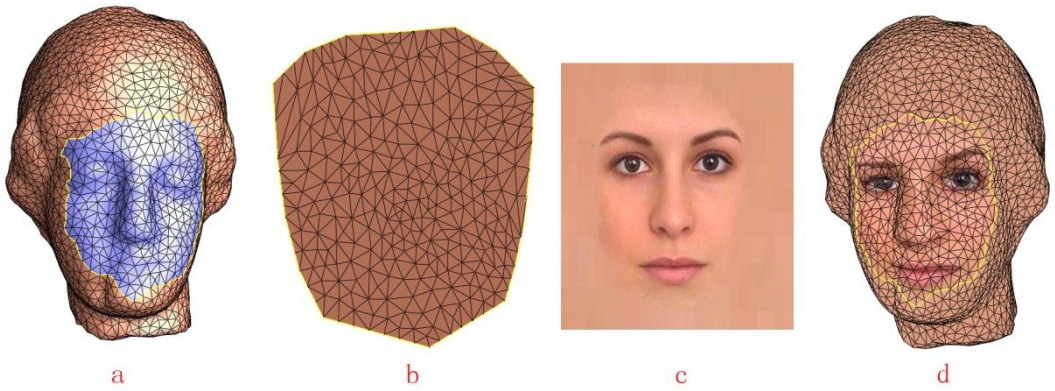

Figure 2.1. Texture mapping as one application of parameterization (harmonic mapping).
Highlighted region of a 3D model in (a) is parameterized to a planar polygon (b). (c) is a 2D image. Texture (c) is mapped to highlighted region through parameterization in (d).

The parameterization is used to put the surface into one-to-one correspondence with an image, stored in the 2D domain. It is possible to map an existing image onto the 3D model, or to define the parametric space image by directly painting the model.

Another class of applications is remeshing. The coordinate system defined by the parameterization facilitates transforming from a mesh representation into another one. This is of great importance for modeling and simulation tasks, which use representations that are completely different from the dense triangulated meshes constructed from 3D scanners and their bundled reconstruction software. Figure 2.2 (in [**7**]) illustrates remeshing from a triangular mesh Chinese Horse (not shown in the figure) to a quad-mesh via a polycube mapping. We can see in the zoom-in region that the elements of the mesh are regular squares rather than triangles.

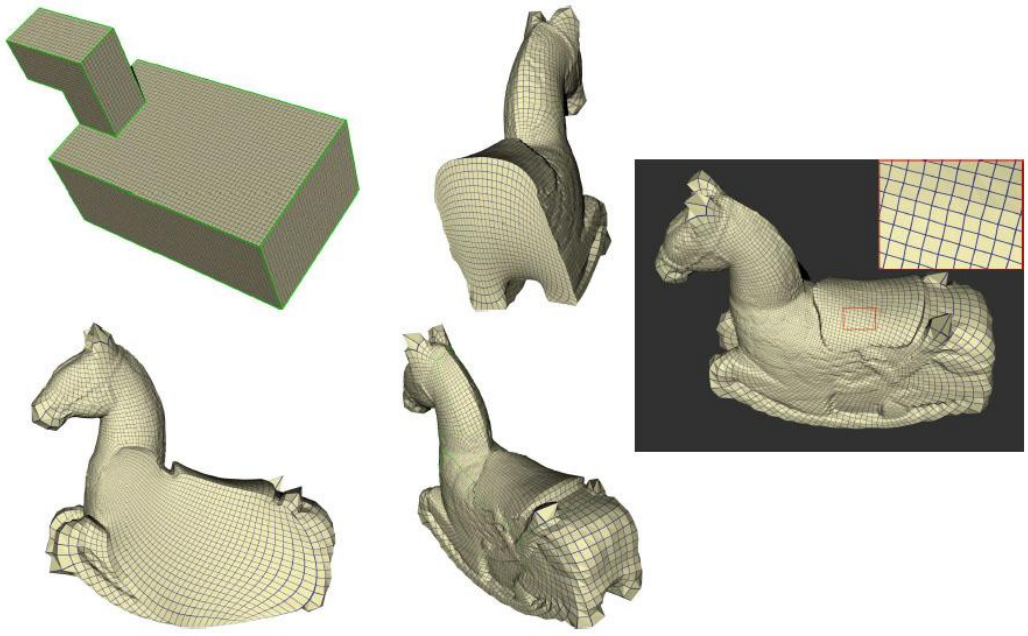

Figure 2.2. Remeshing Chinese horse
It is via a polycube mapping with input as triangular mesh (not shown) and output as quad-mesh.

Generally, surfaces with complicated topology, especially high genus surface (whose genus> 0), are first decomposed to patches, which is a topological disc. Then these patches

are related by parameterizations on planar domain. However, there exists discontinuity along the cutting boundary. Therefore surfaces are preferred to map to the domain with the same genus. Regular shapes, including polycubes, provide a more natural domain without cutting for the closed surfaces whose genus is zero.

In summary, a parameterization of a 3D surface is a mapping putting a surface in one-to-one correspondence with a 2D domain. This notion plays an important role in geometry processing since it makes it possible to transform complex 3D modeling problem into a 2D space where problems are simpler to solve. However, this transformation always introduces some distortion, which will make this reduction from 3D space to 2D space inaccurate. The next sub-chapters discretize the notion of parameterization into a context of a piecewise linear triangle mesh. Then surface parameterization over regular domains, i.e. polycubes, will be presented. At the end of this chapter, the measurement of distortion will be introduced.

2.2 Triangular Mesh Representation

Before introducing the notion of parameterization of surfaces, the representation of surfaces in triangle mesh has to be presented first. Let us denote points in $\mathbb{R}^3$ by $\boldsymbol{p} = (x, y, z)$ and points in $\mathbb{R}^2$ by $\boldsymbol{u} = (u, v)$. An *edge* is then defined as the convex hull of (or equivalently the line segment between) two distinct points and a *triangle or face* as the convex hull of three non-collinear points.

A mesh S_M is defined as $S_M = (V, E, F)$, where $V = \{\boldsymbol{p}_1, \cdots, \boldsymbol{p}_{n+b}\}$ is a set of vertices, $\mathrm{E} = \{E_1, \cdots, E_l\}$ is a set of edges, and $\mathrm{F} = \{F_1, \cdots, F_m\}$ is a set of faces. In vertex set V, $\boldsymbol{p}_1, \cdots, \boldsymbol{p}_n$ are the interior points, and $\boldsymbol{p}_{n+1}, \cdots, \boldsymbol{p}_{n+b}$ are boundary points, if any. A mesh is called triangular mesh, if all the faces in F are triangles. Figure 2.3 is a triangle mesh model Bunny.

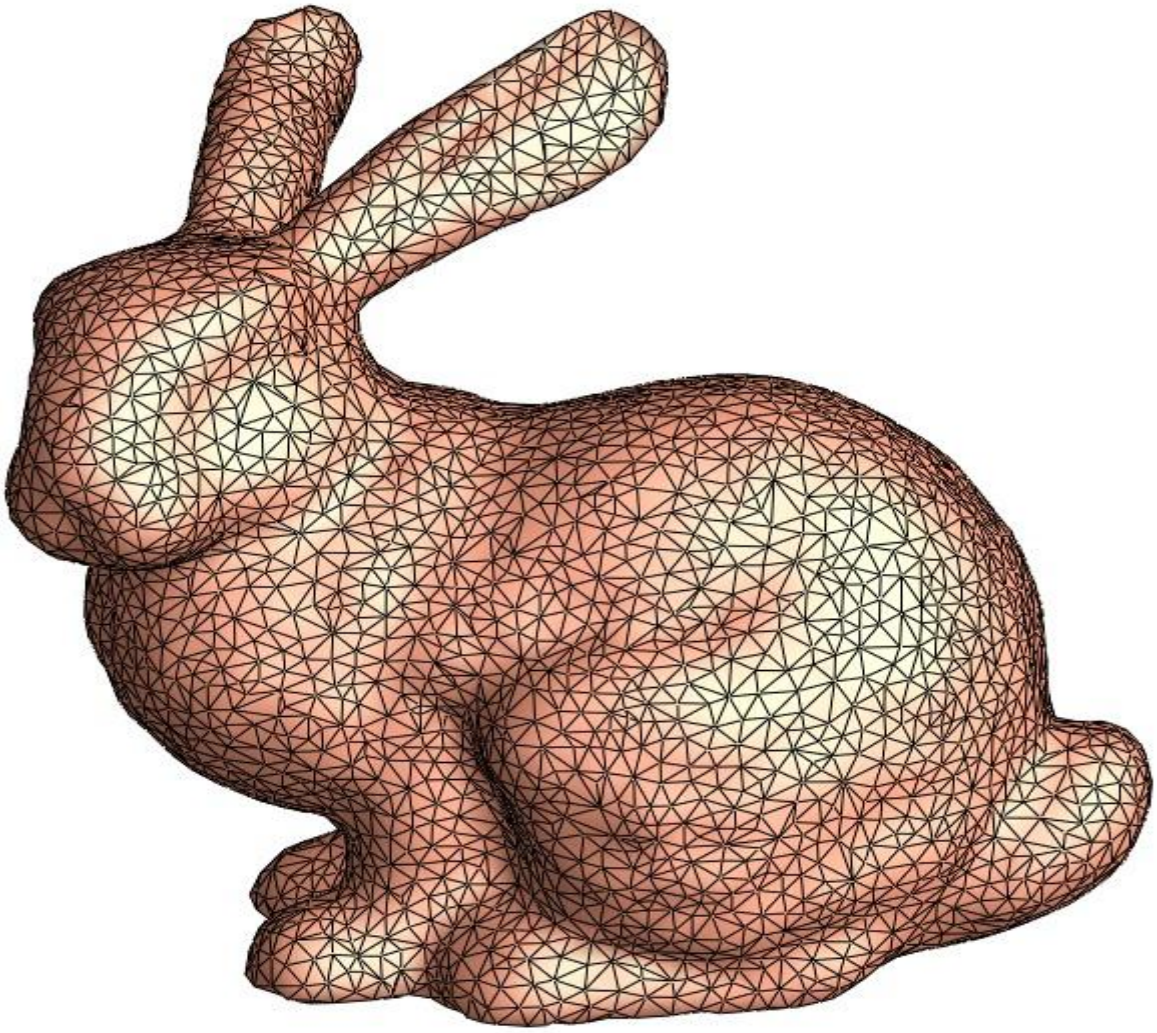

Figure 2.3. Triangle mesh model Bunny.

Each triangle face is a linear segment of a piecewise linear surface representation. Every point $\boldsymbol{p}$ in the interior of a triangle $[\boldsymbol{p}_0, \boldsymbol{p}_1, \boldsymbol{p}_2]$ can be written as a barycentric combination [**21**] of the corner points:

$$\boldsymbol{p} = \alpha \boldsymbol{p}_0 + \beta \boldsymbol{p}_1 + \boldsymbol{p}_2 \qquad with \quad \alpha + \beta + \gamma = 1$$

Therefore, a mapping from a 3D triangle $T = [\boldsymbol{p}_0, \boldsymbol{p}_1, \boldsymbol{p}_2]$ to a 2D triangle $t = [\boldsymbol{u}_0, \boldsymbol{u}_1, \boldsymbol{u}_2]$ as in Figure 2.4, where $\boldsymbol{p}_i \in \mathbb{R}^3, \boldsymbol{u}_i \in \mathbb{R}^2$, can be defined as a linear transformation $f: T \to t$. This is also a trivial case of surface parameterization.

The ideal parameterization, which is distortion-free mapping, is called isometric mapping. Isometric mapping preserves the length of the edge, preserving both the angle and

the area. Therefore, the angle distortion and area distortion vanish in isometric mapping. However, as is well known, this ideal mapping only exists in very special cases. Therefore, researchers have been trying to either minimize the angle distortion or the area distortion, or some combination of angle and area distortion. The parameterization free of angle distortion is called conformal mapping, and the one free of area distortion is called equiareal mapping. For more details, we refer to [**9**] for a good survey.

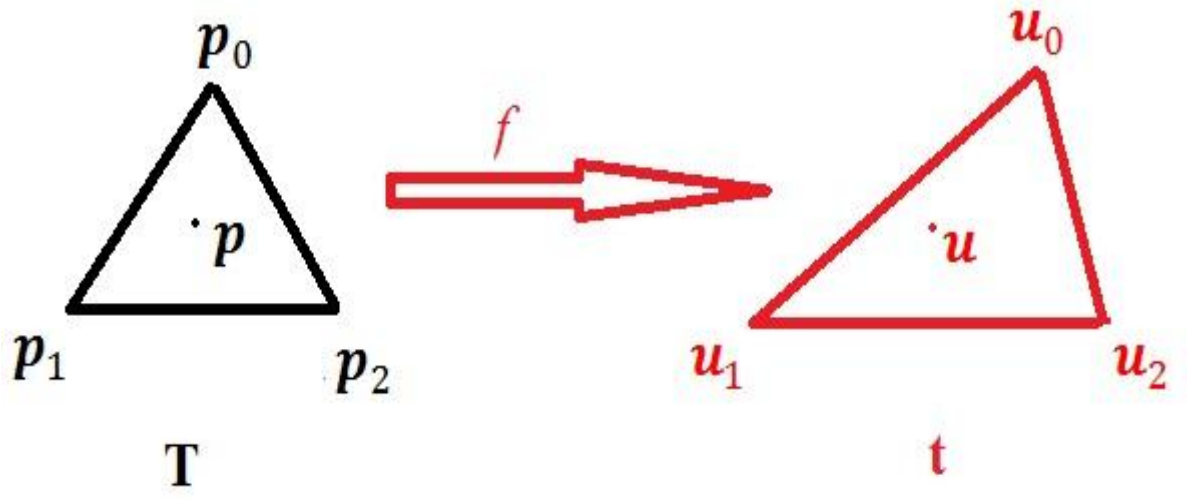

Figure 2.4. Triangle map f from 3D triangle T to 2D triangle t.

2.3 Parameterization of Surface Patches on Planar Domain

Usually, a parameterization f of triangle mesh S_T is uniquely determined by specifying the *parameter points* $\boldsymbol{u_i} = f(\boldsymbol{p_i})$ for each vertex $\boldsymbol{p_i} \in V$ and requiring that f is continuous and linear for each triangle. In this setting, $t = f(T)$ is the linear triangle map from a 3D surface triangle $\mathrm{T} = \{\boldsymbol{p_i}, \boldsymbol{p_j}, \boldsymbol{p_k}\}$ to *parameter triangle* $\mathrm{t} = \{\boldsymbol{u_i}, \mathbf{u_j}, \boldsymbol{u_k}\}$, see Figure 2.6. The parameter domain Ω is the union of all parameter triangles. Sometimes, the parameter triangle need not be planar, therefore the parameterization f is not necessarily a linear map, e.g. the parameter triangle t could be a spherical triangle as in Figure 2.5.

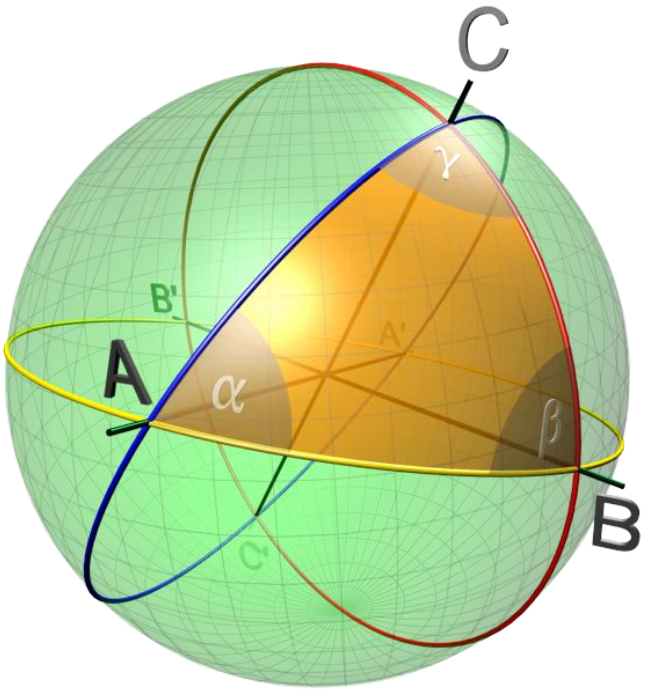

Figure 2.5. Spherical triangle (Courtesy Wikipedia).

A simple idea for establishing a parameterization of a triangle mesh, which is a topological disc (i.e. genus = 0 and there is one boundary), is based on the following physical model. Imagine that the edges of the triangle mesh are springs connected at the vertices. If we fix the boundary of this spring network somewhere in the plane, then the interior network will relax in the energetically most efficient configuration, and we can simply assign the positions where the joints of the network have come to rest as parameter points.

$$E = \frac{1}{2}\sum_{i=1}^{n}\sum_{j\in N_i}\frac{1}{2}D_{ij} \parallel \boldsymbol{u}_i - \boldsymbol{u}_j \parallel^2,$$

where $D_{ij} = D_{ji}$ is the spring constant of the spring between $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$, with respect to the unknown parameter position $\boldsymbol{u}_i = (u_i, v_j)$ in triangle t for the interior points and $N_\mathrm{i} = \{j: edge\ [\boldsymbol{p}_i, \boldsymbol{p}_j]\ exists\}$. The partial derivative of E with respect to $\boldsymbol{u}_i$ is

$$\frac{\partial E}{\partial \boldsymbol{u}_i} = \sum_{i\in N_i} D_{ij}(\boldsymbol{u}_i - \boldsymbol{u}_j),$$

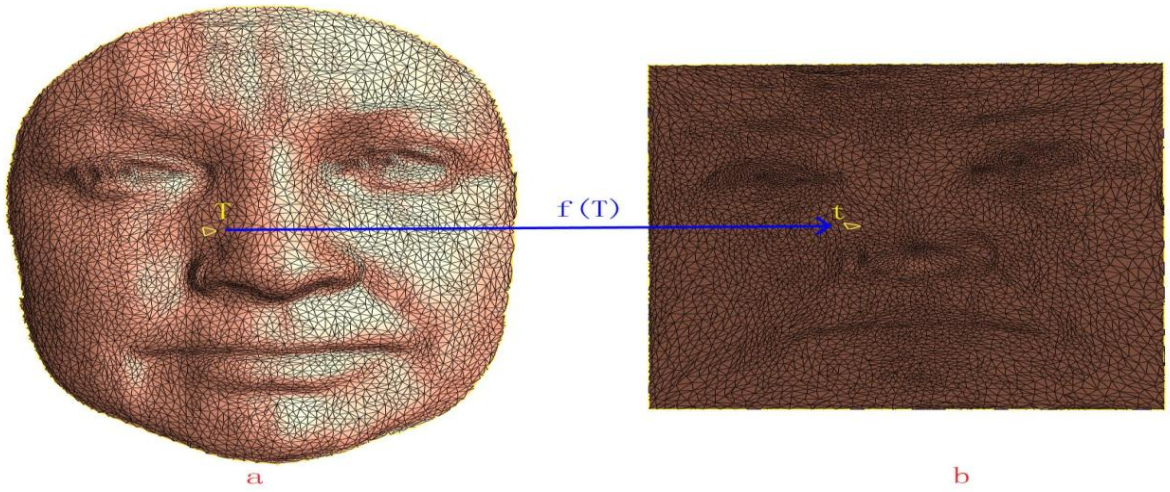

Figure 2.6. Parameterization of a triangle mesh, a topological disc. (a) is a 3D mesh as input. (b) is the parameterized mesh in 2D space.

and the minimum of E is obtained if

$$\sum_{j \in N_i} D_{ij} \boldsymbol{u}_i = \sum_{j \in N_i} D_{ij} \boldsymbol{u}_j$$

Holds for all $i = 1, \cdots, n$. This is equivalent to saying that each interior parameter point $\boldsymbol{u_i}$ is an *affine combination* of its neighbors,

$$\boldsymbol{u}_i = \sum_{j \in N_i} \boldsymbol{\lambda}_{ij} \boldsymbol{u}_j$$

Equation 1

with normalized coefficients

$$\boldsymbol{\lambda}_{\mathrm{ij}} = \frac{\boldsymbol{D}_{ij}}{\sum_{k \in N_i} \boldsymbol{D}_{ik}}$$

Equation 2

that obviously sum to 1.

By separating the parameter points for the interior and the boundary vertices in the sum on the right hand side of Equation 2 we get

$$\boldsymbol{u}_i - \sum_{j \in N_i, j \leq n} \lambda_{ij} \boldsymbol{u}_j = \sum_{j \in N_i, j > n} \lambda_{ij} \boldsymbol{u}_j$$

and see that computing the coordinates u_i and v_i of the interior parameter points $\boldsymbol{u}_i$ requires to solve the linear equations

$$\boldsymbol{AU} = \bar{\boldsymbol{U}} \;\; and \;\; \boldsymbol{AV} = \bar{\boldsymbol{V}},$$

Equation 3

Where $U = (u_1, \cdots, u_n)$ and $V = (v_1, \cdots, v_n)$ are the column vectors of unknown coordinates, $\bar{U} = (\bar{u}_1, \cdots, \bar{u}_n)$ and $\bar{V} = (\bar{v}_1, \cdots, \bar{v}_n)$ are the column vectors of with coefficients

$$\bar{u}_i = \sum_{j \in N_i, j > n} \lambda_{ij} u_j \;\; and \;\; \bar{v}_i = \sum_{j \in N_i, j > n} \lambda_{ij} v_j$$

and $A = (a_{ij})$, where $i = 1, \cdots, n$ and $j = 1, \cdots, n$, is the $n \times n$ matrix with elements

$$a_{ij} = \begin{cases} 1 & if\ i = j \\ -\lambda_{ij} & if\ j \in N_i \\ 0 & otherwise \end{cases}$$

Methods for efficiently solving these systems include direct solvers and iterative solvers, e.g. Cholesky decomposition and conjugate gradient approach [**10**] [**22**].

2.3.1 Barycentric Mapping

The question remains how to choose the spring constants D_{ij} in the spring model, or more generally, the normalized coefficients λ_{ij} in Equation 1. The most naïve choice of these spring constants is $D_{ij} = 1$, which is called barycentric mapping. It is one of the most widely used methods to establish a parameterization of triangular meshes. This method could date back to 1960's, based on Tutte's Barycentric mapping theorem in graph theory [**21**].

Floater applied this idea to construct parameterizations [**23**]. The idea consists of first fixing the vertices on the boundary of a convex polygon. Then the coordinates at the internal vertices are found by solving Equation 3. Another choice for the spring constant is that $D_{ij} = -|N_i|$, where N_i denotes the number of one-ring neighbors of vertex i (i.e. its valence).

However, these weights do not take the mesh geometry into consideration, such as angles and edge lengths, and therefore introduce large distortions not favored by most applications. For this reason, the next sub-chapter presents a way to minimize some distortions.

2.3.2 Conformal and Harmonic Mappings

Conformal mapping is related to formalism of complex analysis. It plays a particular role in complex and Riemannian geometry and has many nice properties. For the moment, consider the case of mappings from a planar region S to the plane. Such a mapping can be viewed as a function of a complex variable, $\omega = f(z)$. Locally, a conformal map is simply any function f, which is analytic in a neighborhood of a point z, and such that $f'(z) \neq 0$. A conformal mapping f satisfies the Cauchy-Riemann equations, with $z = x + iy$ as well as $\omega = u + iv$, which are

$$\frac{\partial v}{\partial x} = -\frac{\partial u}{\partial y}, \qquad \frac{\partial v}{\partial y} = \frac{\partial u}{\partial x}.$$

Equation 4

Now notice that by differentiating these equations with respect to x and y, we obtain two Laplace equations

$$\Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0, \qquad \Delta v = \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} = 0$$

Any mapping $(u(x,y), v(x,y))$ which satisfies these two Laplace equations is called *harmonic mapping*. Therefore a conformal mapping is also harmonic and we have the implications

Isometric ⇒ conformal ⇒ harmonic

The big advantage of harmonic mapping over conformal mapping is that the former is easier to compute, at least approximately. After choosing a suitable boundary mapping, the harmonic mapping can be obtained by solving a system of linear equations.

Surface parameterization on planar domain is a well-studied field. Since isometric mapping only exists for developable surfaces, i.e. surface with zero Gaussian Curvature. Applications show different tolerance to the distortions, including angle distortion, area distortion and stretch. Table 2-1 gives a summary of recent literature regarding the distortions, boundary freedom, bijectivity and optimization complexity.

2.4 Parameterization of Closed Surface over Regular Domains

Lots of parameterization applications require inter-surface mapping between different models. Pair-wise mapping between models can be employed to transfer different properties, including textures, deformation and animation. Blending and morphing, as well as mesh completion and repair, can also utilize pair-wise mapping. The most common way for pair-wise mapping is to parameterize both objects on a common base domain. Parameterizations over planar domain usually require cutting in the models and introduce discontinuities and large distortions along the cutting boundaries. Therefore, it is worthwhile to parameterize the closed surface to a domain with the same genus. Regular domains including polycubes and spheres provide natural base domains for the closed surfaces due to their regularity and simplicity. In this sub-chapter, recent work on surface parameterization of closed surface over regular domains, including polycubes and spheres, will be reviewed.

2.4.1 Polycube Mapping

As a useful parametric domain, polycube maps have been studied in many different shape modeling applications. Tarini et al. invented the concept of polycube map and applied it to the texture mapping and synthesis [**1**]. Fan et al. extended it to generate cross parameterization and morphing by mapping surfaces to polycubes then composing the map by finding the correspondence between them [**3**]. In these approaches, polycube maps are computed by extrinsic methods such as projections. Wang et al. introduced an intrinsic

method for polycube maps and built splines representation on the polycube parametric domain [**4**]. Compared with extrinsic methods, the intrinsic approach reduced the mapping distortion significantly.

Table 2-1. Summary of parameterizations on planar domain.
LSCM: Least Square Conformal Mapping; DCP: Discrete Conformal Parameterization ;ABF: Angle Based Flattening; MIPS: Most Isometric Parameterizations; MDS: Multi-Dimensional Scaling.

Method	Distortion minimized	Boundary	Bijectivity	Complexity
Uniform [21**]**	None	Fixed, convex	Yes	Linear
Harmonic [13**]**	Angles	Fixed, convex	No	Linear
Shape preserving [23**]**	Angles	Fixed, convex	Yes	Linear
Mean-value [15**]**	Angles	Fixed, convex	Yes	Linear
LSCM/DCP [14**] [**24**]**	Angles& Area	Free	No	Linear
ABF/ABF++ [25**] [**26**]**	Angles	Free	Locally no flips	Nonlinear
Linear ABF [27**]**	Angles	Free	Locally no flips	Linear
MIPS [28**]**	Angles	Free	Yes	Nonlinear
Circle patterns [29**]**	Angles	Free	Locally no flips	Nonlinear
Stretching minimizing [30**]**	Distance	Free	Yes	Nonlinear
MDS [31**]**	Distance	Free	No	Nonlinear
Degener et al. [32**]**	Areas	Free	Yes	Nonlinear

Later, Wang et al. developed user controllable polycube maps for manifold spline construction [**5**]. Both approaches required much user involvement in polycube design. Lin at al. presented an automatic polycube mapping approach [**16**], but the bijectivity was not guaranteed. Recently, He et al. presented a divide-and-conquer approach for automatic polycube map construction [**17**]. In that paper, the bijectivity was guaranteed and the

mapping had shown low angle and area distortion. Han et al. applied volumetric polycube maps to construct hexahedral shell mesh [**18**].

Xia et al. introduces an editable polycube mapping [**19**], based on a divide-and-conquer strategy, which gives much more control over the quality of the induced subdivision surface and makes processing of large models with complex geometry and topology feasible.

2.4.2 Spherical Mapping

Spherical mapping is the parameterization $f: S \to S^*$, where the target surface S^* is a unit sphere. The big advantage of the sphere domain over the planar one is that it allows for seamless, continuous parameterization of genus-0 models, e.g. brains and etc. Meanwhile, an important fact is that, according to [**33**], harmonic mapping $f: M \to S$, is equivalent to conformal mapping if M is a genus-zero surface and S is a unit sphere. This means that harmonic mapping dealing with sphere-like surface are free of angle distortion. Haker et al. first parameterizes the given genus-zero surface onto the plane using harmonic mapping after cutting the input surface by using one triangle as a boundary [**34**]. Then they employ *stereographic projection*, a conformal mapping, to map the result to the sphere. The choice of the boundary triangle heavily affects the result. This approach works quite well in practice, but there is no theoretical guarantee since the stereographic projection is bijective on in continuous case, and can produce flip-overs in the discrete case [**10**]. Later, Gu and Yau avoided cutting the sphere by an Gauss-Seidel iterations approximating a harmonic map (and therefore conformal) [**35**]. They considered a mapping solving the non-linear equations

$$\sum_{j \in N(\boldsymbol{v}_i)} w_{ij} \left(\Pi_{\boldsymbol{v}_i}(f(\boldsymbol{v}_j)) - f(\boldsymbol{v}_i) \right) = 0, \qquad \boldsymbol{v}_i, \boldsymbol{v}_j \in V$$

where $N(\boldsymbol{v}_i) = \{\boldsymbol{v}_j | \, edge\,(\boldsymbol{v}_i, \boldsymbol{v}_j) \in E\}$ and $\Pi_{\boldsymbol{v}_i}(u)$ denote the perpendicular projection of any point $\boldsymbol{u}$ on the unit sphere. They began with an initial guess and then moved one vertex at

a time, first computing the Euclidean coordinates for it using the barycentric coordinates [**13**], and then projecting the vertex to the unit sphere. Many other researchers, as in [**27**] [**37**] [**38**], also had used similar strategy based on Gauss-Seidel iterative extension of planar barycentric methods. Unfortunately, Saba et al. [**39**] later proved that projected Gauss-Seidel iterations decrease the residual for only a finite number of iterations. After approaching a bijective solution, the residual increases and the system collapses to a degenerate solution. In other words, this method is unstable.

Afterwards, a method involving linear equations was proposed by Gotsman et al. [**40**] generalizing planar barycentric mapping to spherical case. A theorem by Colin de Verdiere described in this paper guarantees a valid spherical embedding if certain conditions hold for the coefficient matrix of the linear equations. Later, Saba et al. [**39**] provided a scheme to efficiently solve this system, by providing a good initial guess and a robust solver.

An efficient and bijective alternative is suggested by multi-resolution techniques. These methods obtain an initial guess by simplifying the model until it becomes a tetrahedron, trivially embed it on the sphere, and then progressively insert the vertices until the mesh recovers original size [**41**] [**42**]. Praun and Hoppe [**42**] alternated the refining progressively with local relaxation of single vertex position, which minimizes the stretch metric of the parameterization and maintains a valid embedding.

From a different perspective, Sheffer et al. [**43**] generalized the planar angle-based parameterization to spheres using combined angle and area distortion from the planar version in [**26**]. This paper tried to minimize a quadratic energy function subject to non-linear constraints. It is very time-consuming the solve the optimization, which makes their approach only work on very small models. As a regular domain, polycube has its own advantages over other regular domains like spheres due to its planarity of its facets. In this thesis, we will focus on the polycube domains.

2.5 Distortion Metrics

As a piecewise linear mapping, surface parameterization over planar domain introduces some distortion to angles and areas, which is unfavorable for applications. However, the ideal parameterization, i.e. isometric mapping, is very rare and only exists in very special cases, e.g. mapping from a bounded planar region to a cylinder. Researchers explore different ways to minimize these distortions.

Pinkall and Polthier in [**12**] and Eck et al. in [**13**] considered the Dirichlet energy of a given mapping g

$$E_D(g) = \frac{1}{2}\int \|\nabla g\|^2$$

as a measure of deformation. Considering a linear atomic map $f: f(\boldsymbol{x}) = A\boldsymbol{x} + \boldsymbol{b}$, as in Figure 2.4, the discrete Dirichlet energy is the $Frobenius\ Norm$ of the coefficient matrix A

$$\kappa_F(f) = \kappa_F(A) = \frac{\cot\alpha\,|a|^2 + \cot\beta\,|b|^2 + \cot\gamma\,|c|^2}{4Area(t)}$$

where α, β, γ are the angles in T and a, b, c are the edge length in t. For derivations and details, we refer to [**28**]. $\kappa_F(f)$ can also be viewed as the discrete harmonic energy on one triangle.

For area distortion on the atomic map f, it is directly formulated as

$$E_{area}(f) = \frac{Area(T)}{Area(t)} + \frac{Area(t)}{Area(T)}$$

considering the penalty for very large or tiny parameter triangle t. Obviously, for an isometric mapping, the value $\kappa_F(f) = E_{area}(f) = 1$. However, many other distortion metrics, such as $L^2 - norm$ in [**30**], are used in different applications. For more details, please refer to [**22**].

2.6 Summary of Literature Review

As described above, the distortions almost always exist in surface parameterizations. Many distortion metrics and the optimization algorithms to minimize the distortion have been proposed by the researchers. As a result, different surface parameterization approaches have been proposed. A detailed coverage of the surface parameterization techniques and formulation of the distortion metrics is out of the scope of this thesis. We refer to [**9**] [**10**] [**22**] for excellent surveys on this topic.

Chapter 3 Polycube Mapping and Optimization

A polycube domain $\boldsymbol{P}$ is composed of a set of rectangular patches $\{\boldsymbol{P_i}\}$. A polycube map is therefore composed of a set of rectangle maps. In the chapter, we shall use the harmonicity and area distortion to measure the mapping quality and optimize the domain shape as well as the mapping.

3.1 Overview

Ideally, given a metric, we shall simultaneously optimize the polycube domain $\boldsymbol{P}$ as well as the mapping $\boldsymbol{f}: \boldsymbol{S} \rightarrow \boldsymbol{P}$ to minimize the distortion $\boldsymbol{E}(\boldsymbol{f})$. We can formulate this as minimizing $\boldsymbol{E}(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{E}(\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_{3n}, \boldsymbol{y}_1, \boldsymbol{y}_2, \dots, \boldsymbol{y}_{3n})$, with the constraints that $(\boldsymbol{x}_{3i-2}, \boldsymbol{x}_{3i-1}, \boldsymbol{x}_{3i})$ is a point on $\boldsymbol{S}$, and $(\boldsymbol{y}_{3i-2}, \boldsymbol{y}_{3i-1}, \boldsymbol{y}_{3i})$ is the corresponding corner point on the polycube $\boldsymbol{P}$, for $i = 1, \dots, n$.

Directly solving this nonlinear optimization is highly expensive. As will be discussed shortly in Chapter 3.4 Optimization, the derivatives of $\boldsymbol{E}$ over $\boldsymbol{y}$ can be computed efficiently, but the derivatives of $\boldsymbol{E}$ over $\boldsymbol{x}$ could not be computed in practice. Without derivatives of the object function, this optimization with complicated constraints is difficult even for moderately large $\boldsymbol{n}$. To make full use of the partial derivative information of the objective function, we iteratively do the optimization over $\boldsymbol{x}$ (for optimal polycube corner mapping) and $\boldsymbol{y}$ (for optimal polycube domain shape) separately. Hence, gradient based nonlinear optimization methods using the derivatives of $\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{y}}$ can be developed to efficiently optimize the sub-problem $\boldsymbol{E}(\boldsymbol{x}, \boldsymbol{y})$ for fixed $\boldsymbol{x}$. Meanwhile, a derivative-free optimization algorithm is developed to optimize the sub-problem $\boldsymbol{E}(\boldsymbol{x}, \boldsymbol{y})$ for fixed $\boldsymbol{y}$. During each iteration, when the shape of every rectangle and the mappings of its four corner points are determined, we can

compute/update the mapping efficiently (see Chapter 3.3 Mapping and Chapter 3.4 Optimization). The proposed iterative polycube mapping optimization framework therefore has the following three steps (illustrated in Figure 3.1).

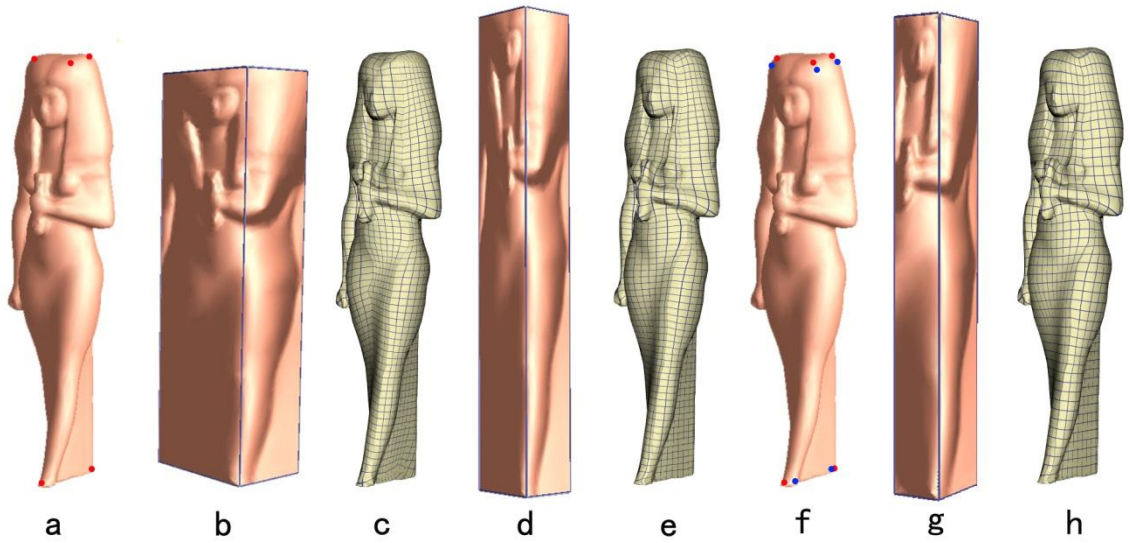

Figure 3.1. Algorithm overview.
(a) original surface with eight corner points(red). (b)(c) initial polycube domain and mapping. (d)(e) optimized polycube domain and mapping. The harmonic energy with area distortion term is reduced from 5.4414 to 4.7812. (f) the optimized polycube mapping with eight new corner points(blue) with a lower harmonic energy of 4.5961. (g)(h) final optimized domain and optimized mapping after two iterations. The grid quality is improved.

1. Initial Polycube Domain Construction (Chapter 3.2 Polycube Construction). Given a budget number of corner points, an initial polycube domain is constructed either automatically or manually, meeting the corner point budgets; then the corner point mapping and the initial polycube mapping are computed.

2. Optimizing Polycube Domain Shapes (Chapter 3.4 Optimization). Preserving the topology of the polycube, the scaling of sub-patches is optimized so that mapping energy is minimized.

3. Optimizing Polycube Mapping (Chapter 3.4 Optimization).

Table 3-1. Algorithm for optimal polycube mapping

Algorithm 1: Optimal Polycube Mapping.
Input : surface S , corner point number n;
Output : polycube mapping $f : S \rightarrow P$;
1. Construct an initial polycube P_0, whose corner point number $\leq n$;
2. Compute an initial mapping $f_i: S \rightarrow P_i$; $i = 0$;
3.Repeat
4.$i \leftarrow i + 1$;
5.Optimize the polycube domain P_i , s.t. distortion of mapping f_{i-1} is minimized;
6.Optimize the polycube map $f_i: S \rightarrow P_i$;
7.until $\lvert P_i - P_{i-1} \rvert < \epsilon$;
8.Perform a global smoothing.

The framework is formulated in Table 3-1. Note that in our iterative process, we keep on optimizing scaling factors of sub-patches and the corner points. Then (1) *polycube domain optimization* takes corner points decided by the current mapping $\boldsymbol{f}_i$ as the input and solve scaling of sub-patches to reduce mapping distortion; and (2) *polycube mapping optimization* uses the scaled polycube $\boldsymbol{P}_{i+1}$ as the target domain and optimizes the location of corner points. This iterative refinement converges when the polycube domain shape $\boldsymbol{P}_i$ does not change any longer.

3.2 Polycube Construction

The initial polycube can be constructed manually [**1**] [**4**] or automatically [**16**] [**17**]. We also use a simple voxelization algorithm (Chapter 3.2.1) to generate the polycube. Since this initial polycube and maps (Chapter 3.3) will be optimized to minimize the distortion, a simple, efficient, and adaptive (to different corner budgets) scheme such as this voxelization

algorithm is sometimes enough. The following optimization framework is general, and can optimize an initial polycube mapping constructed via different methods.

3.2.1 Construct Polycube via Voxelization

Given a solid object $\boldsymbol{M}$, supposing its boundary surface is represented by a triangle mesh $\boldsymbol{S} = (\boldsymbol{V}_s, \boldsymbol{E}_s, \boldsymbol{F}_s)$ where $\boldsymbol{V}_s, \boldsymbol{E}_s, \boldsymbol{F}_s$ V are vertex, edge, and face sets, we construct a polycube domain $\boldsymbol{P} = (\boldsymbol{V}_p, \boldsymbol{E}_p, \boldsymbol{F}_p)$, and corresponding corner points mapping using a voxelization algorithm. Figure 3.2 illustrates a polycube construction example of a Buddha model through voxelization.

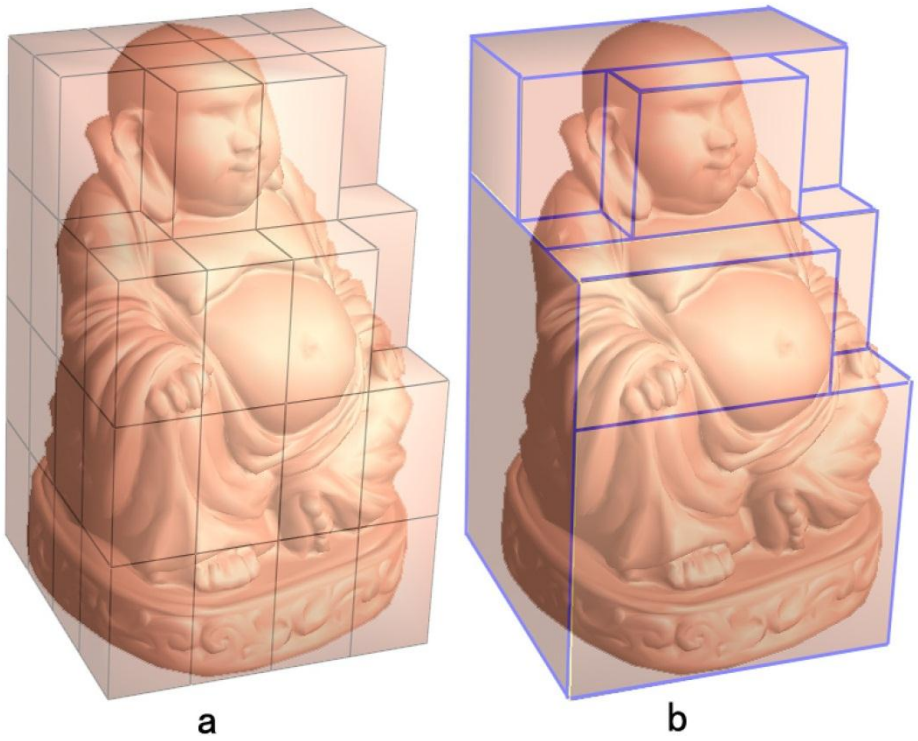

Figure 3.2. Voxelization for polycube construction.

We use an octree to represent the object. The subdivision starts from a rectangular bounding box. Each cell (rectangular cuboid) can be labeled as *inside* or *outside*. Then we remove all interior faces that are shared by two inside cells, and finally merge all inside cells to one polycube $\boldsymbol{P}$. The remaining faces form the boundary surface of $\boldsymbol{P}$. We further merge

these remaining faces to a set of big rectangle facets of the polycube. Iteratively, we merge two adjacent faces if the result remains a planar convex polygon. After merging, only rectangle facets are left. The vertices of these rectangles are called corner points, denoted as $\boldsymbol{V_{CP}}$. And the edges of the rectangles form the connectivity of the corner points $\boldsymbol{E_{CP}}$. For each corner $\boldsymbol{v} \in \boldsymbol{V_{CP}}$, we use the simple projection method [**1**] to find its corresponding points on $\boldsymbol{S}$. Without ambiguity, we also call these corresponding points *corner points* on $\boldsymbol{S}$, denoted as $\boldsymbol{V_{CS}}$; they will be mapped to corners in the initial polycube mapping. The voxelization algorithm is simple, automatic and efficient. Moreover, the octree's depth can be adaptively decided by the number of corner points.

Voxelization approaches sometimes provide unnecessary zigzagged domain shapes when the geometry of the object is not well aligned with principal axes, which can be undesirable. Then other polycube domain construction algorithms (e.g. [**1**] [**4**] [**16**] [**17**] may be used to construct the initial mapping, and our subsequent optimization paradigm can still be applied to refine the domain shape and improve mapping quality.

3.3 Mapping

Given the initial polycube $\boldsymbol{P}$, corner point correspondences $\boldsymbol{V_{CS}}, \boldsymbol{V_{CP}}$, and cube edges $\boldsymbol{E_{CP}}$, we compute an initial polycube mapping $\boldsymbol{f}: \boldsymbol{S} \rightarrow \boldsymbol{P}$ as follows. Denote the position of each vertex $\boldsymbol{v}$ on $\boldsymbol{S}$ as $\boldsymbol{X} = (\boldsymbol{x^0}, \boldsymbol{x^1}, \boldsymbol{x^2})$ and its image on the polycube as $\boldsymbol{U} = \boldsymbol{f}(\boldsymbol{X}) = (\boldsymbol{u^0}, \boldsymbol{u^1}, \boldsymbol{u^2}) \in \boldsymbol{P}$; also denote three components of the vector function $\boldsymbol{f}$ as $\boldsymbol{f^0}, \boldsymbol{f^1}$, and $\boldsymbol{f^2}$.

A discrete harmonic parameterization [**13**] is a bijective map from $\boldsymbol{S}$ to a 2D (u,v)-domain, $\boldsymbol{h}: \boldsymbol{S} \rightarrow \boldsymbol{D}, \boldsymbol{S} \subset \mathbb{R}^{\boldsymbol{3}}, \boldsymbol{D} \subset \mathbb{R}^2$ such that the discrete harmonic energies of both $\boldsymbol{u}$ and $\boldsymbol{v}$ components are minimized. When the target planar domain $\boldsymbol{D}$ is convex, and a diffeomorphic boundary mapping is given, the harmonic mapping $\boldsymbol{h}$ is bijective. Therefore,

we can decompose $\boldsymbol{S}$ to multiple patches, each of which will be mapped to a rectangle facet $\boldsymbol{P_i}$ on $\boldsymbol{P}$.

The harmonic energy of a mapping function on t-th (t=0,1,2) component is defined as

$$H^t = \frac{1}{2}\sum_i \sum_{v_j \in N(v_i)} w_{ij}\left(f^t(\boldsymbol{X}_i) - f^t(\boldsymbol{X}_j)\right)^2$$

Equation 3.5

where $N(v_i)$ is the set of all 1-ring neighboring vertices of $\boldsymbol{v_i}$. $w_{ij} = \frac{1}{2}(\cot\alpha_{ij} + \cot\beta_{ij})$ is the well-known cotangent weight [**13**] defined on the edge $[\boldsymbol{v}_i, \boldsymbol{v}_j] \in \boldsymbol{E}_S$, where $\boldsymbol{\alpha}_{ij}$ and β_{ij} are two angles opposite to the edge $[\boldsymbol{v}_i, \boldsymbol{v}_j]$.

For each polycube edge in $[\boldsymbol{v_{pi}}, \boldsymbol{v_{pj}}] \in \boldsymbol{E_{CP}}, \boldsymbol{v_{pi}}, \boldsymbol{v_{pj}} \in \boldsymbol{V_{CP}}$, we trace curves to connect their corresponding points $\boldsymbol{v_{si}}, \boldsymbol{v_{sj}} \in \boldsymbol{V_{CS}}$ using shortest paths following algorithms introduced in [**44**]. After this, the harmonic mapping computation is straightforward.

We parameterize these traced paths to polycube edges using the arc-length parameterization. On each facet of the polycube, corner and edge mapping decides the boundary condition and the interior mapping can be computed by solving two sparse linear systems [**13**].

3.4 Optimization

Following Algorithm in Table 3-1, after constructing the initial polycube mapping, we are going to iteratively optimize the polycube domain shape and the polycube mapping.

3.4.1 Polycube Domain Optimization

Given a polycube mapping $f: S \to P = f_i: S_i \to P_i$ defined on a set of topological rectangle patches on S. We want to find the optimal re-scaled P_i so that mapping distortion is

minimized. We use a distortion energy E composed of the harmonic energies $H^t(f), t = 0,1,2$ and an area-stretching term $A(f)$.

$$H^t = \sum_{P_k} H_k^t = \sum_{P_k} \Big(\sum_{e_{ij} \in P_k} \frac{1}{2} w_{ij} \left(f^t(\boldsymbol{X}_i) - f^t(\boldsymbol{X}_j) \right)^2 \Big) ;$$

Equation 3.6

$$A = \sum_{P_k} \sum_{F_{i,j,k} \in P_k} \frac{\left(\Delta(\boldsymbol{U_i}, \boldsymbol{U_j}, \boldsymbol{U_h}) \right)^2}{\Delta(\boldsymbol{X_i}, \boldsymbol{X_j}, \boldsymbol{X_h})} ;$$

Equation 3.7

$$E = H^0 + H^1 + H^2 + \alpha A ;$$

Equation 3.8

where $\Delta(\boldsymbol{X_i}, \boldsymbol{X_j}, \boldsymbol{X_h})$ and $\Delta(\boldsymbol{U_i}, \boldsymbol{U_j}, \boldsymbol{U_h})$ denote the original area of triangle (v_i, v_j, v_h) and the area of its image under the mapping; P_k is a facet of polycube and $F_{i,j,h}$ is a triangle on this facet; α is a weighting factor balancing the harmonic and area-stretching terms.

When optimizing the polycube shape, we restrict our re-scaling on P_i such that (1) it preserves the total area of the polycube, and (2) it doesn't increase the number of corner points. Specifically, we divide the polycube P into different rectangular facets in each coordinate plane (see Figure 3.2).

First, we sort the coordinates of all corner points in three axes, and denote them as $\{ \alpha_j^i \}, i = 0,1,2, j = 0, \dots, N_i$ .We translate the left-bottom of the polycube to the Origin, so that any $\alpha_0^i = 0$.

Then supposing a facet P_k is perpendicular to the u^t coordinate axis, we (1) denote the coordinate of P_k in u^t axis as α_j^t, and (2) on each patch perpendicular to u^t, denote its corresponding coordinates as $[\alpha_j^{t+1}, \alpha_{j+1}^{t+1}]$ and $[\alpha_j^{t+2}, \alpha_{j+1}^{t+2}]$. The superscript indicates the

corresponding axis (u^0, u^1, or u^2), so $t+1$ actually denotes $(t+1)\ mod\ 3$. In our following derivations, the addition of superscripts denotes their addition $modulo\ 3$.

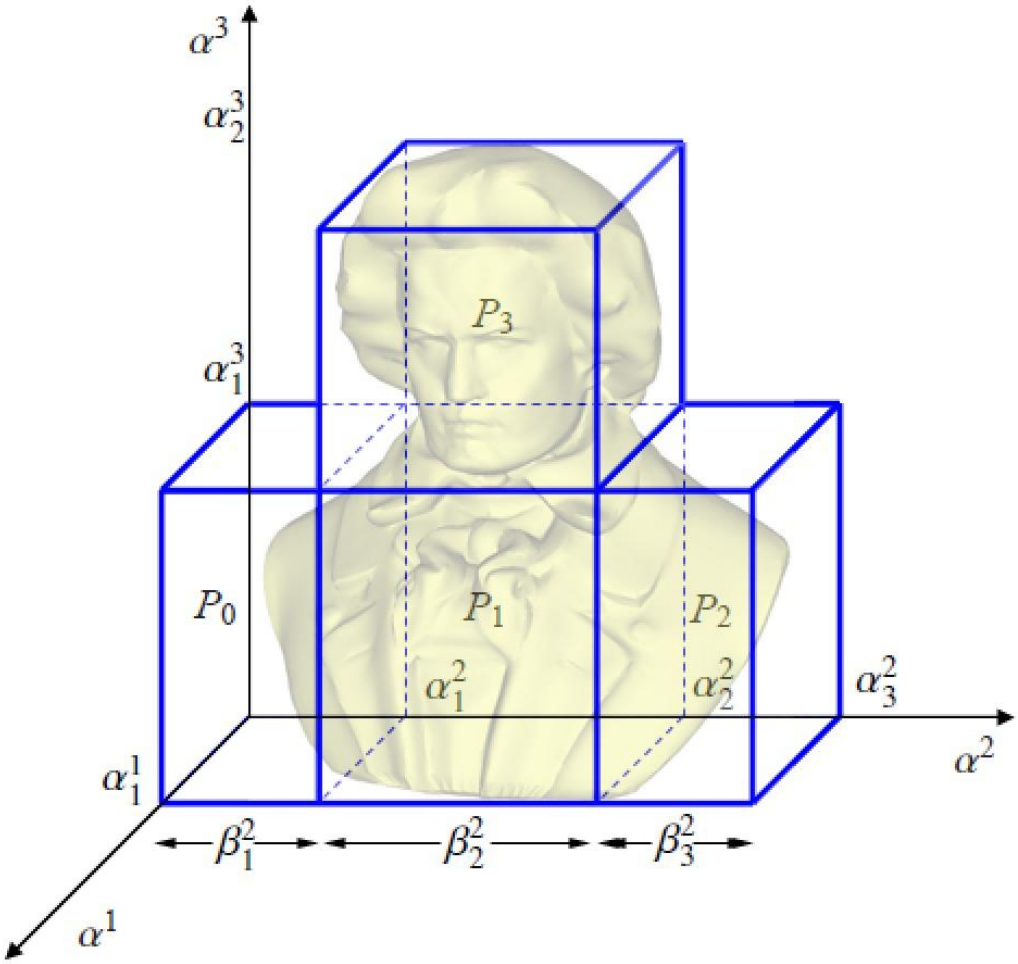

Figure 3.2. Definition of polycube coordinates and parameters.

Now we can denote the length of each segment in $u^t - axis$ as $\beta_{j+1}^t = \alpha_{j+1}^t - \alpha_j^t$; and adjacent facets (faces connected by a same polycube edge) should share a same corresponding scaling factor β, to prevent the increase of corner points.

Therefore, supposing a rectangle domain P_k is perpendicular to the axis $u^i, (i = 0,1,2)$, we denote the two corresponding segment lengths of the rectangle as $\beta^{i+1}(P_k)$, $\beta^{i+2}(P_k)$ their initial lengths as $\widetilde{\beta^{i+1}}(P_k)$, $\widetilde{\beta^{i+2}}(P_k)$, initial harmonic energies as $\widetilde{H^{i+1}}(P_k)$, $\widetilde{H^{i+2}}(P_k)$, and initial area stretching energy as $\widetilde{A_{P_k}}$. These constants $\widetilde{\beta^{i+1}}(P_k)$, $\widetilde{\beta^{i+2}}(P_k)$, $\widetilde{H^{i+1}}(P_k)$, $\widetilde{H^{i+2}}(P_k)$, $\widetilde{A_{P_k}}$ are determined by the initial mapping. Then the harmonic energy

of all sub-patches that are perpendicular to u^i, with respect to the their scalings can be written as:

$$E_H^i(\alpha_1^i, \dots, \alpha_{N_i-1}^i, \beta_1^i, \dots, \beta_{N_i-1}^i) = \sum_{P_k} ((\beta^{i+1}(P_k))^2 \widetilde{C_k^{i+1}} + (\beta^{i+2}(P_k))^2 \widetilde{C_k^{i+2}})$$

Equation 3.9

where $\widetilde{C_k^{i+1}}$ and $\widetilde{C_k^{i+2}}$ are constants decided by the initial mapping:

$$\widetilde{C_k^{i+1}} = \left(\frac{\widetilde{H_{P_k}^{i+1}}}{(\widetilde{\beta^{i+1}}\ (P_k))^2} \right), \qquad \widetilde{C_k^{i+2}} = \left(\frac{\widetilde{H_{P_k}^{i+2}}}{(\widetilde{\beta^{i+2}}\ (P_k))^2} \right) ;$$

Considering all three axes, the global harmonic energy of the polycube mapping is:

$$E_H\left(\{\alpha_j^i, \beta_j^i\}, \forall i = 0,1,2, j = 1, \dots, N_i - 1\right) = E_H^1\left(\alpha_1^0, \dots, \alpha_{N_1-1}^0, \beta_1^0, \dots, \beta_{N_1-1}^0\right)$$

$$+ E_H^2\left(\alpha_1^1, \dots, \alpha_{N_2-1}^1, \beta_1^1, \dots, \beta_{N_2-1}^1\right)$$

$$+ E_H^3\left(\alpha_1^2, \dots, \alpha_{N_3-1}^2, \beta_1^2, \dots, \beta_{N_3-1}^2\right);$$

Equation 3.10

The area stretching term of the mapping is:

$$E_H^i\left(\{\alpha_j^i, \beta_j^i\}, \forall i = 0,1,2, j = 1, \dots, N_i - 1\right) = \sum_{P_k} \left(\beta^{i+1}(P_k)\, \beta^{i+2}(P_k) \right)^2 \widetilde{C_k}) ;$$

Equation 3.11

where $\widetilde{C_k}$ is a constant decided by the initial mapping:

$$\widetilde{C_k^{i+1}} = \left(\frac{\widetilde{A_{P_k}}}{(\widetilde{\beta^{i+1}}\ (P_k)\widetilde{\beta^{i+2}}(P_k))^2} \right).$$

Finally, we have the entire distortion energy:

$$E\left(\{\alpha_j^i, \beta_j^i\}\right) = \mathrm{E_H} + \mathrm{E_A}$$

Equation 3.12

subject to the constraints:

$$\begin{cases} \alpha_1^i = \beta_1^i \\ \alpha_1^i + \beta_2^i = \alpha_2^i \\ \alpha_2^i + \beta_3^i = \alpha_3^i \\ \dots \\ \beta_j^i > 0, \forall j = 1, \dots, N_i, i = 0,1,2. \\ \sum_{P_k} \beta^{i+1}(P_k)\beta^{i+2}(P_k) = \widetilde{Area} \end{cases}$$

Equation 3.13

where $\widetilde{Area} = \sum_{P_k} \widetilde{\beta^{i_1}}(P_k)\widetilde{\beta^{i_2}}(P_k)$ the last equation preserves the total area of the polycube domain. Figure 3.3 shows an example of an optimized polycube for the Beethoven model based on the initial polycube mapping. The original polycube (b) is re-scaled to (d); as the grid texture mapping visualized, the distortion of the original mapping (a) reduces when the polycube shape changes (f); as in the zoom-in view (e), the angle distortion is smaller than that in (c).

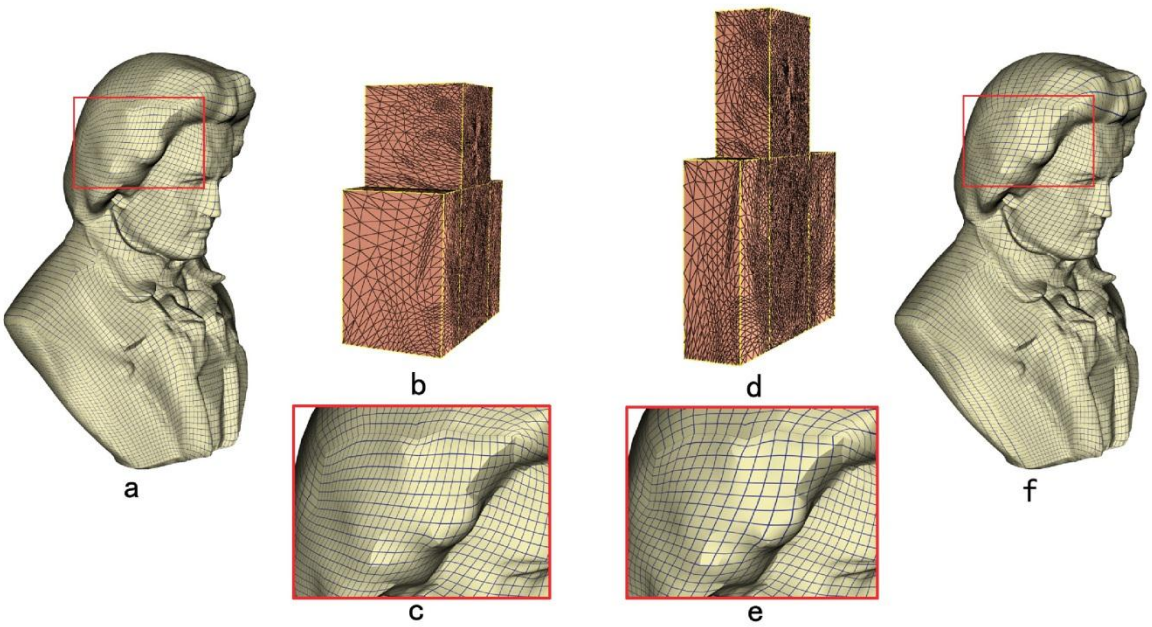

Figure 3.3. Polycube domain optimization.
(a)-(c) shows the initial polycube domain and mapping. (d)-(f) shows the optimized polycube domains. Note the improvement of the checkerboard texture mapping between (c) and (e).

In order to solve the energy $E(\{\alpha_j^i, \beta_j^i\})$ in Equation 3.12 subject to constraints in Equation 3.13, we will strictly enforce all the bounds and linear constraints, and put the last nonlinear constraint as a penalty term $\lambda \left(\Sigma_{P_k} \beta^{i+1}(P_k)\beta^{i+2}(P_k) - \widetilde{Area}\right)^2$ in the objective function. As a result, this optimization problem could be formulated as minimization of a nonlinear function with bound and linear constraints, i.e.,

$$\min \quad E(\boldsymbol{x}), s.t. \quad \boldsymbol{x} \in \Omega \coloneqq \{\boldsymbol{x}: A\boldsymbol{x} = \boldsymbol{b}, \boldsymbol{b_1} \leq \boldsymbol{x} \leq \boldsymbol{b_u}\}$$

Equation 3.14

where $\boldsymbol{x} \in \mathbb{R}^{\boldsymbol{n}}$ is the vector of variables $\{\alpha_j^i, \beta_j^i\}, n = 2(N_1 + N_2 + N_3)$, $\boldsymbol{b_1}$ and $\boldsymbol{b_u} \in R^n$ are the bound constraints, and A is an m by n matrix with $\boldsymbol{b} \in \mathbb{R}^{\boldsymbol{m}}$ denoting the linear constraints. Although the objective function is continuously differentiable, the dimension n of our reformulated problem generally can be large, and the explicit computation of the Hessian is difficult. Hence, first order method, which only requires gradient information, is preferred. To solve Equation 3.14, we employ the following non-monotone gradient projection algorithm, which is also an iterative algorithm: given the starting $\boldsymbol{x}_0$, our algorithm takes the following iterations

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k$$

Equation 3.15

where k is the iteration number, α_k is a stepsize and $\boldsymbol{d}_k$ is the searching direction defined as

$$\boldsymbol{d}_k = P_\Omega\left(\boldsymbol{x}_k - \frac{1}{\lambda_k^{BB}}\boldsymbol{g}_k\right) - \boldsymbol{x}_k.$$

Here, P_Ω is the projection on the feasible set Ω, $\boldsymbol{g}_k = \nabla f(\boldsymbol{x}_k)$and $\lambda_k^{BB}, k \geq 1$ is the so called Barzilai-Borwein [**45**] step-size parameter generated by satisfying a quasi-Newton property,

$$\lambda_k^{BB} = \ argmin_{\lambda \geq \lambda_0} \|\Lambda(\lambda)\boldsymbol{s}_{k-1} - \boldsymbol{y}_{k-1}\|_2$$

Equation 3.16

where $\boldsymbol{s}_{k-1} = \boldsymbol{x}_k - \boldsymbol{x}_{k-1}$, $\boldsymbol{y}_{k-1} = \boldsymbol{g}_k - \boldsymbol{g}_{k-1}$, $\Lambda(\lambda_k) = \lambda_k \boldsymbol{I}$, and λ_0 is a positive constant. Hence, the proposed λ_k^{BB}, when $k \geq 1$, obtained from Equation 3.16, is

$$\lambda_k^{BB} = \max\left\{\frac{\boldsymbol{s}_{k-1}^T \boldsymbol{y}_{k-1}}{\boldsymbol{s}_{k-1}^T \boldsymbol{s}_{k-1}}, \lambda_0\right\}$$

Equation 3.17

and λ_0^{BB} can be arbitrarily defined as a positive number and we set $\lambda_0^{BB} = \|\boldsymbol{g}(\boldsymbol{x}_0)\|_\infty$ and $\lambda_0 = 10^{-10}$ in practice. This BB initial step-size (Equation 3.17) has been extensively studied recently and been shown to perform much better than steepest descent type gradient projection methods [**46**] [**47**]. However, to maintain the efficiency, the step-size α_k in (Equation 3.15) must be obtained by a non-monotone line search. In our experiments, we use the non-monotone line search developed in [**48**] [**49**] .

3.4.2 Polycube Mapping Optimization

In previous subsection, we fix the corner point mapping $f(V_{CS}) \rightarrow f(V_{CP})$ to optimize the shape of polycube domain. We further reduce the mapping distortion by moving vertices V_{CS} (without ambiguity, we also call them corner points) over S. Any 2-dimensional manifold S can be parameterized to an atlas $\Omega = \{\Omega_i\}$, and locally any point on S: $(x^1, x^2, x^3) \in S$ can be represented as a 2D coordinate (r^1, r^2) on a local planar chart. We construct local parameterization $g_i: S_i \rightarrow \Omega_i$ by mapping the *C*-ring neighboring regions (in our experiments, we set $C = 20$) of each initial corner point $\in V_{CS}$ to a unit disc Ω_i . Any neighboring points on the domain Ω_i are continuously parameterized. Let N be the number of the corner points $N = |V_{CP}|$. The optimization will be conducted on all charts $\{\Omega_1, \ldots, \Omega_N\}$ simultaneously by searching the optimal N corner points, represented as coordinates $(\{r_1, r_2, \ldots, r_{2N-1}, r_{2N}\})$, where (r_{2k-1}, r_{2k}) corresponds to (r^1, r^2) on chart Ω_k.

This problem is formulated as minimizing the distortion energy E of the map f decided by the corner maps:

$$E(r_1, r_2, \dots, r_{2N}) = H^1(f) + H^2(f) + H^3(f) + A(f),$$

Equation 3.18

the harmonic energies and area stretching of function f are defined following Equation 3.6 and Equation 3.7.

For polycube mapping with N corner points, the dimension of this optimization problem is $2N$. f is determined by these $2N$ parameters, and can be efficiently computed (Chapter 3.4.2.1), but since we need to retrace the shortest paths as the sub-patch boundaries, we do not have the closed form for f or its derivative. Therefore, we use a derivative-free optimizer (Chapter 3.4.2.2) to solve this problem.

As indicated in Algorithm 1, we iteratively perform domain optimization (Chapter 3.4.1) and mapping optimization (this sub-chapter) until the polycube domain does not change. Despite the optimization of both the domain shape and the corner mapping, the angle distortion near the sub-region boundary (e.g. polycube corners, edges) can be large due to the usage of harmonic mapping with fixed boundary. We perform a smoothing process to further reduce the distortion. Smooth transition functions [**50**] can be easily computed between adjacent polycube faces, then parameterization/smoothing can be computed on a flattened domain covering this boundary region. We adopt the smoothing algorithm of [**51**] to refine the map near polycube corner/edge regions.

Figure 3.4 illustrates an iteration of domain mapping optimization on a Beethoven model. Corners in (a) are adjusted to new positions (f). Meanwhile, the mapping distortion energy reduces, which can also be visualized in the zoom-in regions (d,e vs b,c). If we

perform an aforementioned smoothing, the distortion near the boundary region can be further reduced (f,g).

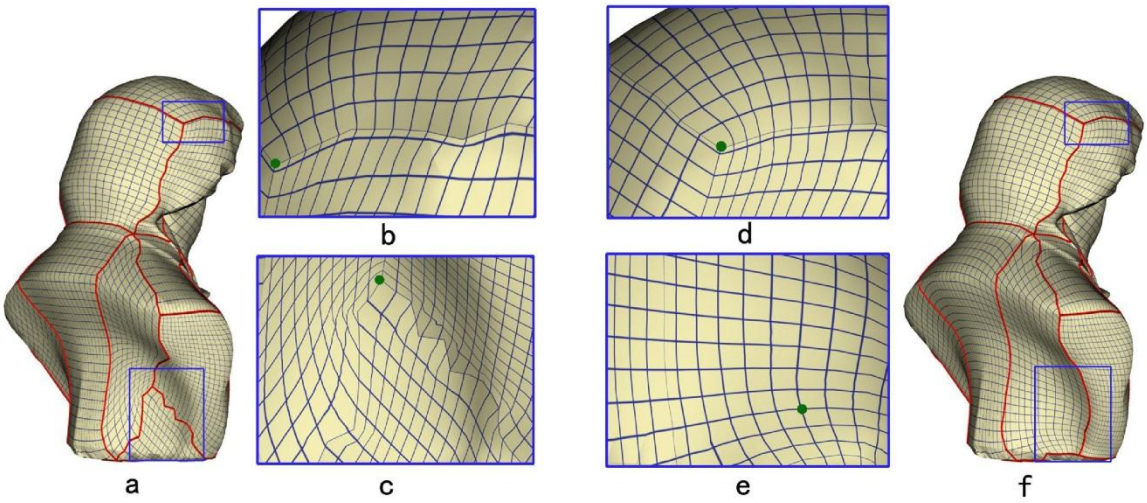

Figure 3.4. Polycube mapping optimization.
(a) is the model before mapping optimization. (b,c) zoom in to show the distortion before this step. (d,e) illustrate the distortion after mapping optimization. (g,f) show distortion after the smoothing post-process. (h) is model after smoothing. The corner points are shown in green. With the smoothing, distortion and discontinuity across sub-region boundaries significantly reduces.

Figure 3.5 shows an iteration of our polycube map optimization on the horse model; the initial horse mapping (a) on a polycube with 60 corner points is optimized; the resultant mapping (b) has smaller angular and area distortion.

3.4.2.1 Efficient Mapping Re-computation

The typical computation for harmonic surface mapping on each rectangle sub-patch involves solving two systems of linear equations. This can be time consuming when we need to re-compute it and re-evaluate its distortion in every step during the optimization. Since the boundary condition of the mapping always changes gradually, we can utilize a more efficient linear equation updating algorithm CHOLMOD [**52**] to accelerate the mapping recomputation.

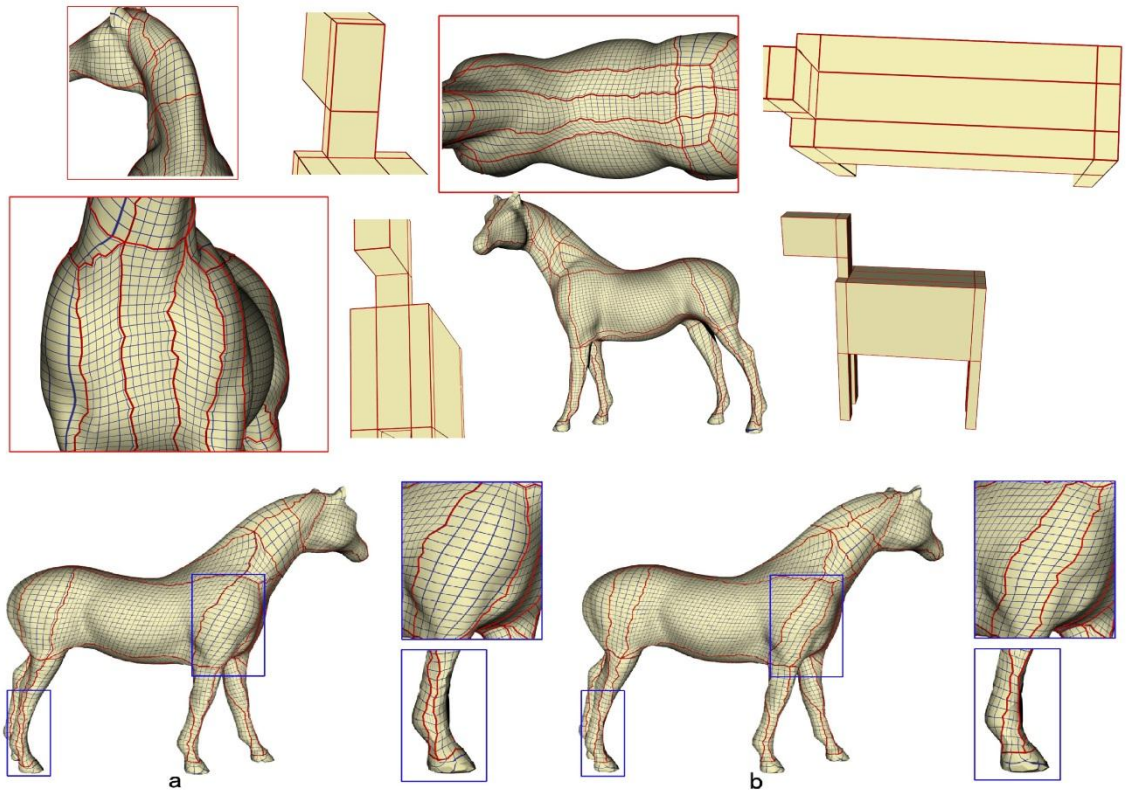

Figure 3.5. Mapping optimization of the Horse model on a polycube
There are 60 corner points. The lower row shows the moving of corner points: (a) before optimization, (b) after optimization.

Mapping on each sub-patch is harmonic, so the coefficient matrix is sparse, symmetric and positive definite. This special property makes it feasible to utilize Cholesky decomposition to solve and update the linear systems very quickly. Initially, we pre-compute the shortest paths between all pairs of vertices using the Floyd-Warshall algorithm and store predecessor matrices on shortest paths. This takes $\boldsymbol{O(n^3)}$ preprocessing time, where $\boldsymbol{n}$ is the number of vertices. During each iteration, when corner points are replaced by some of their neighboring points, between each pair of corners, we retrace corresponding shortest paths in $\boldsymbol{O(k)}$ time where $\boldsymbol{k}$ is the number of vertices on this path. The coefficient matrix only changes slightly (a few rows and columns proportional to the number of mutable boundary conditions due to the change of corner points). This infers an efficient solution-update algorithm. Davis and Hager [**52**] proposed an approach of dynamic supernodal sparse Cholesky update and downdate, which produces a solution for the newly update linear system

without repeatedly computing the coefficient matrix and solving the system. After an initial Cholesky decomposition at a cost of $\boldsymbol{O}(\boldsymbol{n^3})$, the decomposition can be updated in $\boldsymbol{O}(\boldsymbol{N})$, where $\boldsymbol{N}$ is the number of changed entries in Cholesky factor, which is typically much smaller than the size of the mesh, leading to efficient harmonic mapping update. The similar approach was introduced to graphics and shape modeling [**53**] for dynamically updating harmonic fields design.

With this efficient mapping update technique, we can re-evaluate the objective function for a given vector of new planar coordinates for corner points on S. Since the parameterization (and therefore the corner selection) is continuous, we dynamically split each corresponding triangle (where each parametric corner point locates) into three and update the accumulated energy accordingly.

3.4.2.2 Derivative-free Optimization Algorithm

The objective function (**Equation 3.18**) can be reformulated in the following format

$$\min \boldsymbol{\Phi}(\boldsymbol{x}) = \sum_{i=1}^{m} sgn(i) f_i^2(\boldsymbol{x}), \qquad s.t.\ \boldsymbol{b_1} \leq \boldsymbol{x} \leq \boldsymbol{b_u}$$

Equation 3.19

where $\boldsymbol{x} \in \mathbb{R}^{\boldsymbol{n}}, \boldsymbol{b_1}\ and\ \boldsymbol{b_u} \in \mathbb{R}^{\boldsymbol{n}}$ are the bound constraints, and $sgn(i) = \pm 1$ is the sign in front of the squares of $f_i, i = 1, \ldots, m$. The main difficulty of solving this problem is that the explicit derivatives are not available. We develop a trust region based derivative-free algorithm in spirit similar to the approach proposed in [**20**]. Our algorithm does not require the derivative information of the objective function, nor does it explicitly approximate the derivative. Instead, at each iteration, it builds a local quadratic model of the objective function by multivariate interpolation in combination with trust region techniques. More specifically, at each iteration, the algorithm adaptively chooses a set of interpolation points

$\boldsymbol{Y}_k$, with $(n+1) \le |\boldsymbol{Y}_k| \le (n+1)(n+2)/2$, where k is the iteration number and $|\boldsymbol{Y}_k|$ is the cardinality of $\boldsymbol{Y}_k$. Our algorithm takes the following major steps:

Step 0 (Initialization) Set up initial starting guess $\boldsymbol{x}_0$, trust region radius Δ_0 and sampling points $\boldsymbol{Y}_0$. Build initial trust region model on $\boldsymbol{Y}_0$ and set $k = 0$.

Step 1 (Criticality step) Choose a base point $\boldsymbol{y}_k \in \boldsymbol{Y}_k$ and calculate the gradient of our model. If the gradient is sufficiently small, stop. Otherwise, make sure the model is well-posed [**20**] in a trust region with radius proportional to the norm of model gradient.

Step 2 (Step Calculation) Solve the following trust region sub-problem :

$$\min \phi(\boldsymbol{d}), \qquad s.t. \|\boldsymbol{d}\| \le \Delta_k, \ \boldsymbol{l} \le \boldsymbol{x}_k + \boldsymbol{d} \le \boldsymbol{u}$$

Equation 3.20

where $\phi_k(\mathbf{d})$ is a local quadratic model of $\Phi(\boldsymbol{x})$ in a trust region with radius Δ_k. Here, $\|\cdot\|$ is the 2-norm.

Step 3 (Acceptance of the trial step) Compute the ratio of actual and predicted function reduction

$$r_k = \frac{\Phi(\boldsymbol{x}_k) - \Phi(\boldsymbol{x}_k + \boldsymbol{d}_k)}{\phi_k(0) - \phi_k(\boldsymbol{d}_k)},$$

where $\boldsymbol{d}_k$ is the minimizer of (Equation 3.20). If $r_k > 0$, then $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{d}_k$; otherwise, $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$.

Step 4 (Trust region radius update)} Update trust region radius by

$$\Delta_{k+1} = \begin{cases} \frac{1}{2}\|\boldsymbol{d}_k\| & if\ r_k < 0.1 \\ \max\{\frac{1}{2}\Delta_k, \|\boldsymbol{d}_k\|\} & if\ 0.1 \le r_k < 0.7 \\ \max\{\Delta_k, 2\|\boldsymbol{d}_k\|\} & if\ r_k \ge 0.7 \end{cases}$$

If $r_k \ge 0.1$, form $\boldsymbol{Y}_{k+1}$ from $\boldsymbol{Y}_k$ by merging new point $\boldsymbol{x}_{k+1}$. Set $k = k+1$, go to Step 1.

Step 5 (Model improvement) This step applies only when $r_k < 0.1$. In this case, before shrinking the trust region radius, make sure the model is well-posed [**20**] in the current trust region. Set $k = k + 1$, go to Step 1.

One critical advantage of this algorithm is using the least Frobenius norm updating strategy [**48**] to update the quadratic model (Equation 3.20). Hence, to build our quadratic model, we only need $O(n)$ (in our experiments, $2n + 1$) function evaluations, while normally $(n + 1)(n + 2)/2 = O(n^2)$ number of valuations are required for building a fully quadratic model (Note, $(n + 1)(n + 2)/2$ could be much bigger than $2n + 1$ for relatively large n). In addition, at each iteration, only one new function evaluation is required to update the local quadratic model.

Therefore, our approach is usually more efficient [**55**] [**20**] than other widely used strategies in derivative-free optimization, such as using finite-difference to approximate derivatives [**50**] or some direct search methods [**57**]. Global convergence of the algorithm as well as the good local geometry of the set of interpolation points are guaranteed by trust region techniques [**20**] [**48**].

3.5 Applications

We also demonstrate an application of our polycube mapping framework in multiple objects mapping.

3.5.1 Intersurface Mapping among Multiple Objects via Polycube

Polycube can be used as a canonical base domain for multiple objects (preferably, these objects have the same topology and similar geometry). Our framework can be used to generate such a common regular domain, and multiple objects are parameterized onto this single polycube with low distortion.

Multiple shapes can be analyzed, processed, and integrated over this single domain. Supposing we have a set of models $\{ S_1, \dots, S_n \}$ to be integrated, we construct a common polycube P using S_1. We also compute initial mapping f_i between P and each $S_i, i = 1, \dots, n$. Then simultaneously, we optimize P and the mapping $f_i: S_i \rightarrow P$ using the above proposed framework. The final polycube domain P is the one that minimizes the total distortion of multiple polycube parameterization $E = \Sigma_i E(f_i)$. The final polycube is an optimal domain for all these models. Inter-surface mapping between two models S_i and S_j can be composed and optimized over this domain as $f_{i,j}: S_i \rightarrow S_j = f_j^{-1} \circ f_i$. We visualize our optimal polycube and the mapping results using inter-object morphing by linearly interpolating them over the common polycube domain.

Specifically, we construct initial polycube P for S_1 and use projection to determine corner points mapping. However, this simple projection approach does not work well when we map P to other models $S_2, S_3, \dots, S_n$, especially when S_i is not geometrically similar to P. Especially for this situation (when we want to map a surface to a dissimilar polycube), we compute the initial polycube mapping in the following more robust way (Note that any other suitable polycube mapping approach can also be used to generate initial f_i). We partition P and each S_i consistently (i.e. the segmentation of P and S_i has the isomorphic dual graph); then compute the mapping $f_i: S_i \rightarrow P$ by merging all individual sub-region mappings. Such an approach based on canonical pants decomposition is introduced in [**58**]. We briefly recap the basic idea, and refer readers to [**58**] for details. The pants patch is a genus-0 surface with 3 boundaries. Any surface (except for a few trivial cases) can be decomposed into a set of pants patches, including g handle patches and a base patch, where g is the genus. The base patch is then further iteratively partitioned into a set of pants patches.

Finally, every pants patch is decomposed into two sub-patches, each of which can be parameterized on a regular planar hexagon. Therefore, the global surface mapping between two objects is composed by parameterizations of sub-patches on these hexagonal domains. This approach can easily and robustly handle the surface mapping between two objects with arbitrary topology and feature points, therefore it is suitable here for generating initial mapping $f_i: S_i \rightarrow P$. Figure 3.6 shows an example of using the above approach to construct optimal common polycube for the horse and the cow. Individually optimal polycubes for the horse and cow are shown in (b) and (c), and initial polycube maps are visualized in (a) and (d); the optimal common polycube is shown in (f). Specifically, a compromise can be seen in the neck region. The final common polycube mappings are visualized in (e) and (g).

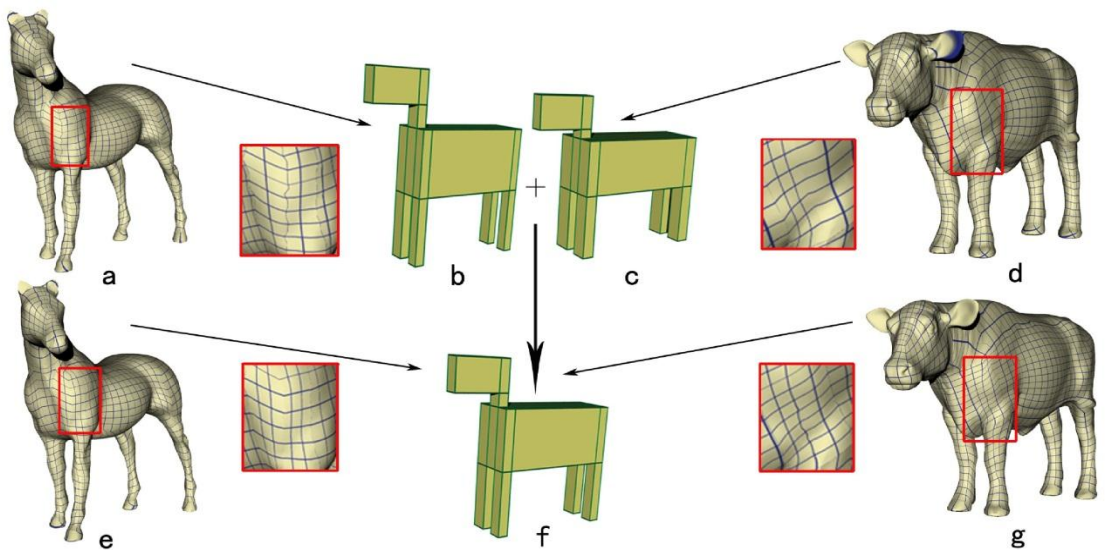

Figure 3.6. Common polycube mapping for multiple models.
Initial polycube maps of the horse and cow are as (a) and (d); individually-optimal polycube domains are shown in (b) and (c); the common optimal polycube domain is shown in (f); and the final common optimal polycube mapping of both models are as (e) and (g). Note: the common polycube balances both individually-optimal polycubes, see the neck region.

3.5.1 Volumetric Polycube Parameterization

Solid volumetric data have richer contents than the surfaces. When the data processing and analysis are related to material, intensity, or any other structural information defined over

the whole 3D region of the objects, we need to consider the shape as 3-manifold and study the volumetric mapping. It is a natural generalization of surface mapping, which facilitates similar applications when solid data are prevalent: matching, meshing, spline construction and etc. Polycube mapping, as the boundary surface mapping of volumetric parameterization, induces the volumetric polycube parameterization, which can be applied to feature alignment and remeshing.

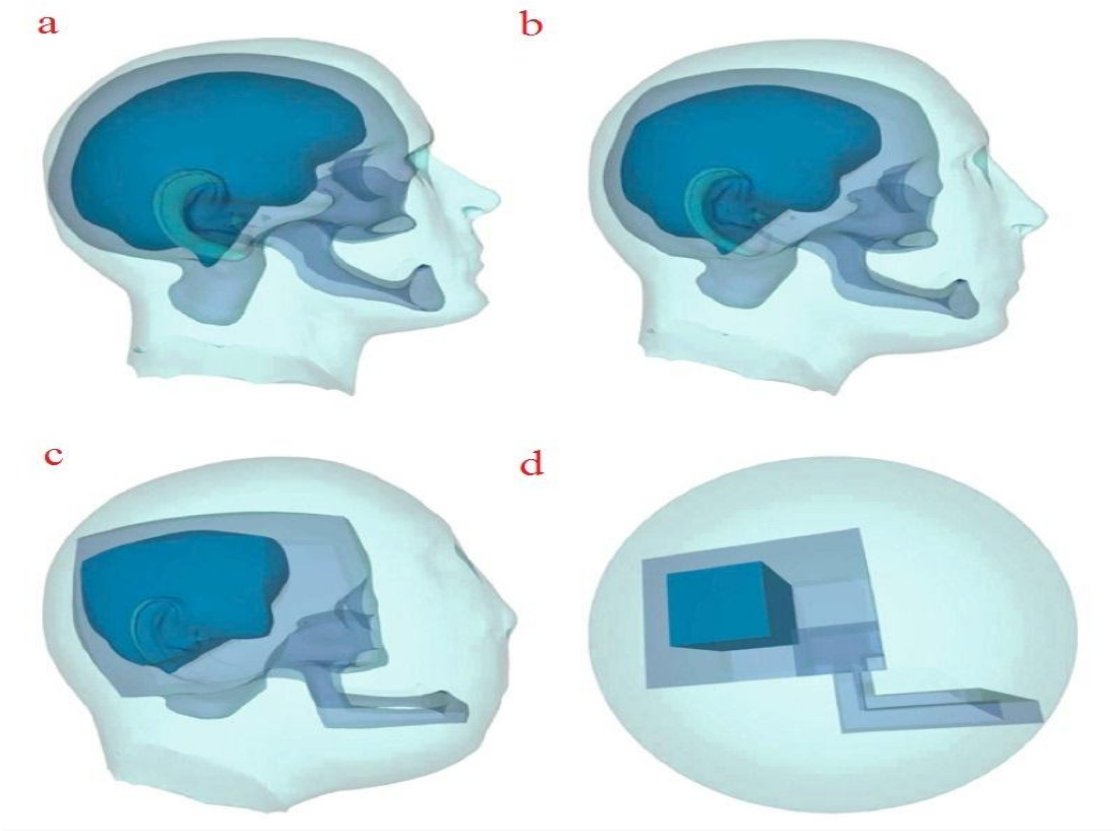

Figure 3.7. Heterogeneous volumetric mapping (boundary surface mapping). This volumetric mapping is between head-skull-brain and polycube-sphere. (a) The extracted and cleaned volumetric shape has three salient iso-surfaces: head, skull, and brain. A target domain (d) is generated to test the efficacy of our mapping with constraints of iso-surface. (d) has a sphere, a polycube skull, and a cube inside, corresponding to three iso-layers in (a). (b) and (c) show the 30% and 60% morphing from (a) to (d) by linear interpolation.

In real scenarios, volumetric data usually contain different materials and densities, or have salient structure inside its interior region (see Figure 3.7a). These information or structures are usually meaningful and should be considered. Therefore, a scheme that can

properly handle heterogeneous structure is worthwhile, so that we will be able to align or match similar material/intensity when necessary.

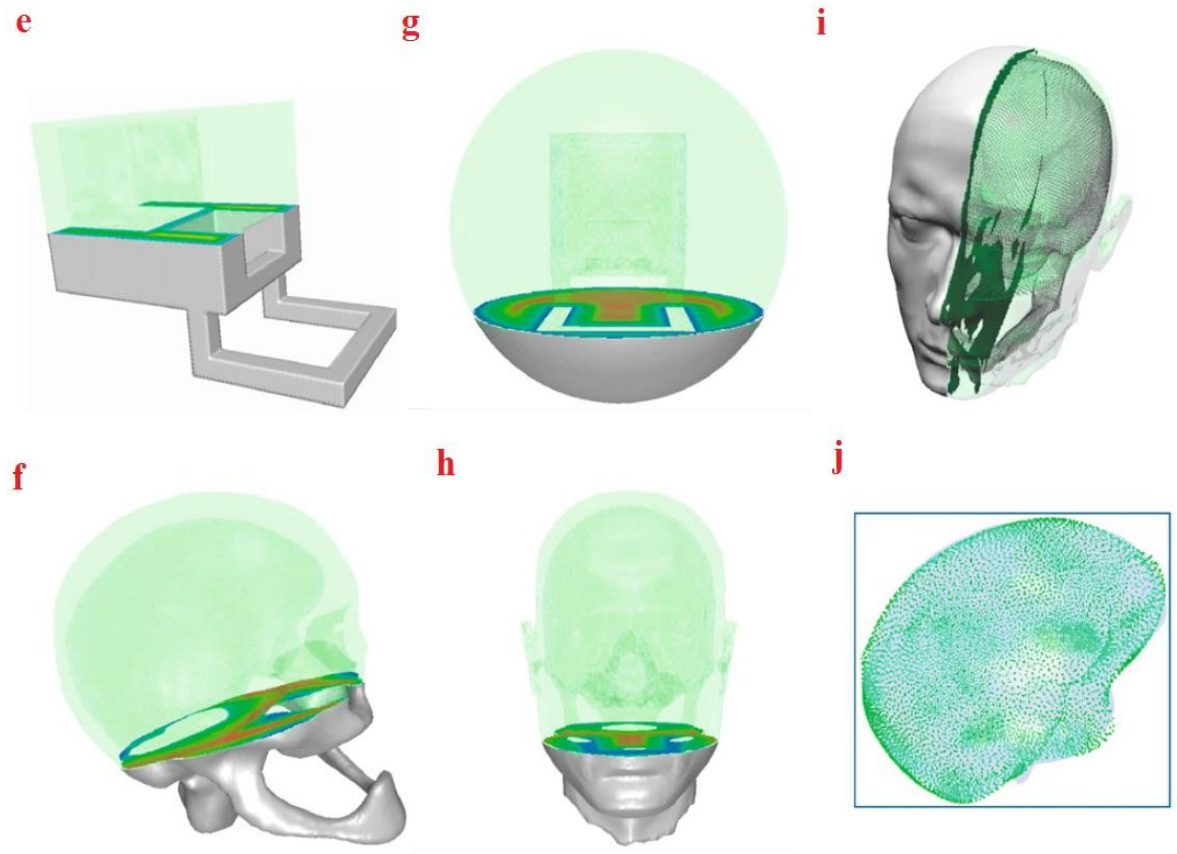

Figure 3.8. Heterogeneous volumetric mapping (volumetric mapping). (e)(g) show cross sections on the polycube-sphere domain. (f)-(i) show corresponding cross-section on the head-skull-brain model. The point clouds in (e)-(i) are the sampled feature points on these iso-surfaces (e)(g), and their images (f)-(i) of volumetric mapping. The color-encoding in (f)(h) visualizes the mapping via transferred distance field of (e)(g). In (j), the brain iso-surface and its fitting, green points are images of sampled points on the interior cube in (e).

Figure 3.7 and Figure 3.8 show an example of a volumetric mapping over the heterogeneous data head-skull-brain model, which has three salient iso-surfaces: the outer boundary is a genus-zero (head) surface, and the interior skull iso-surface is genus-two, within which there is a genus-zero (brain) surface. The polycube domain is generated to test the efficacy of our mapping on heterogeneous 3D data with iso-surface constraints.The outer head boundary surface is mapped onto s sphere boundary, the skull iso-surface is constrained on the polycube skull, while the brain iso-surface is mapped to s small cube inside. Locations

of the feature points in (g-i) demonstrate that the iso-surface constraints are precisely fitted, and the volumetric mapping align the feature surface very well. More volumetric polycube mapping results are in Figure 3.9.

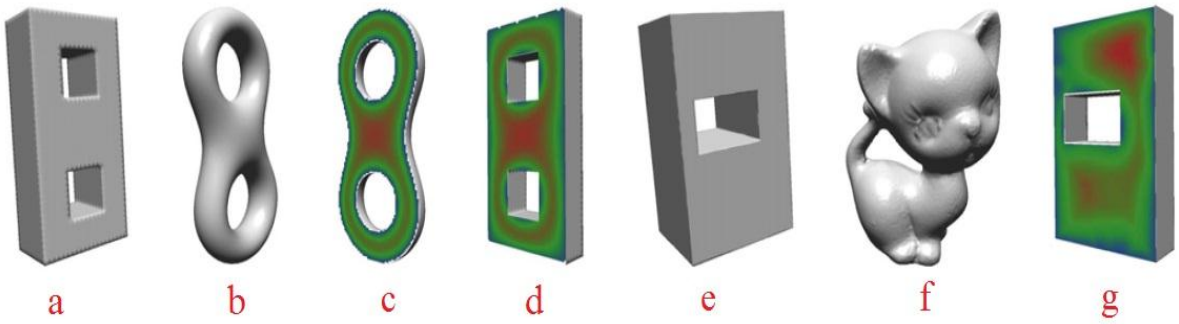

Figure 3.9. Mapping between solid objects and polycubes.
Polycubes (a,e) are mapped to two-torus (b) and kitten (f), respectively. Color-encoded distance field of (c,g) are transferred under the mapping to (d, h).

A direct application for volumetric mapping is hex-mesh generation. Regular mesh structure is highly desirable for finite element and physically based deformation/simulations, because regular meshes provide great efficiency for geometry processing and physically based simulation [**7**]. Given an 3D solid data M, we first compute the polycube mapping $f': \partial M \to \partial P$, where ∂M and ∂P is the boundary surface of M and P, respectively, then volumetric polycube mapping $f: M \to P$. With f we can transfer the regular structure on P to M. Figure 3.10 illustrates an example of using a unit solid cube to remesh the solid David head. We compute the volumetric mapping $f: M_1 \to M_2$ from the cube to the David head. Then $f(M_1)$ is a solid with the hex connectivity of M_1 and the head shape of M_2, and it is the remeshed David head, as illustrated in (c)(d). More hex remeshing examples are also illustrated in Figure 3.11.

3.6 Experimental Results

We compare the property of our polycube mapping framework with existing methods and list them in

Table **3-2**. Our method generates the optimal polycube within the same topological class, and the complexity of the polycube is flexibly bounded by the given number of singularities. We test our optimization framework on a few 3D shapes. Figure 3.12 shows the optimization on Bimba and Max-Planck. The texture-mapped rectangular grids become closer to squares, indicating the reducing of angle distortion.

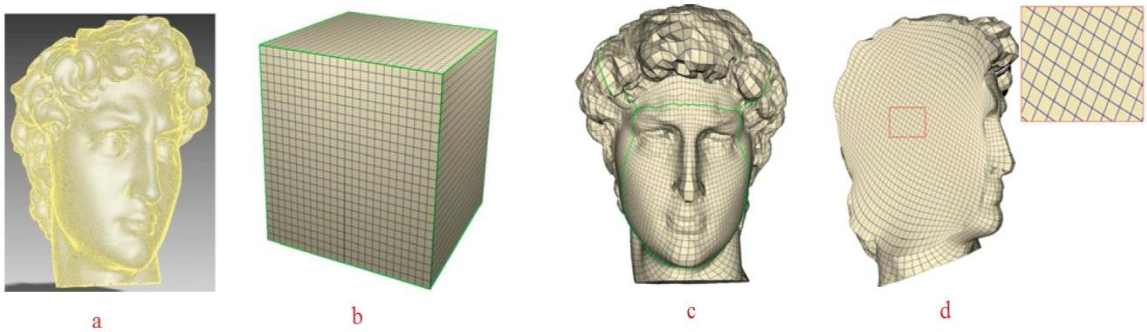

Figure 3.10. Hex remeshing of the solid David head.
(a) The original mesh structure of the David head. (b) A simple cube domain that the hexahedral mesh is generated upon. (c) The remesh David and (d) a cross-section to show the interior structure.

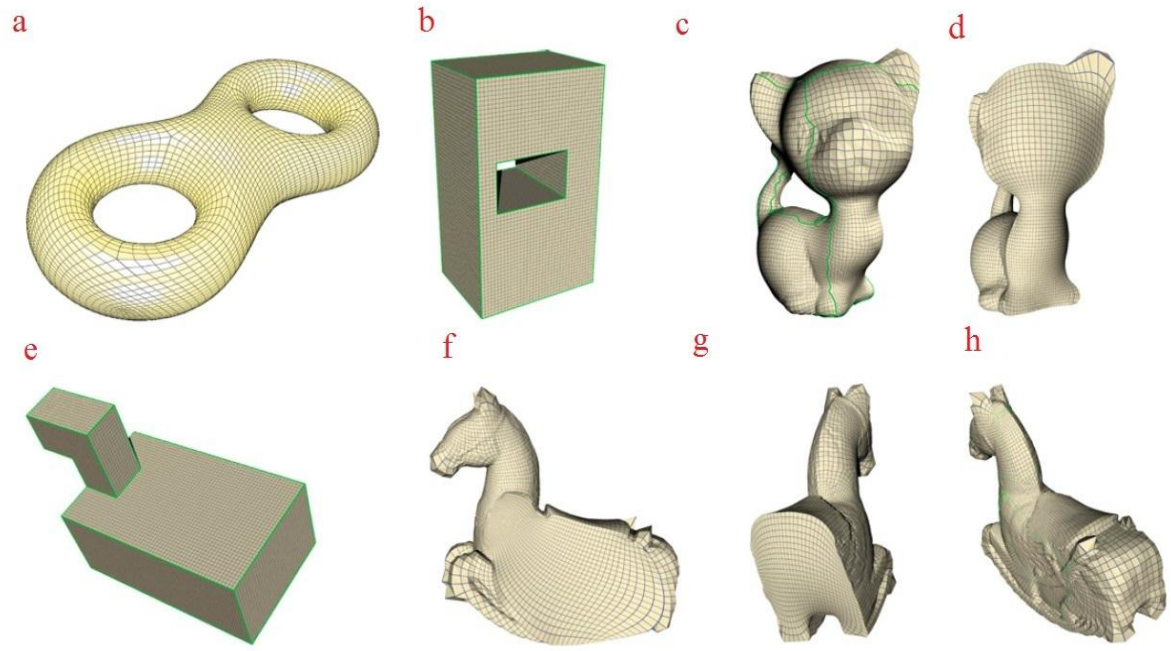

Figure 3.11. Hex remeshing.
(a) illustrates a hex-remeshed solid two-torus. The hex mesh on the polycube for remeshing solid kitten is shown in (b). The remeshed kitten is illustrated in (c)(d). (e)-(h) show the hex-remeshing for a solid Chinese horse model.

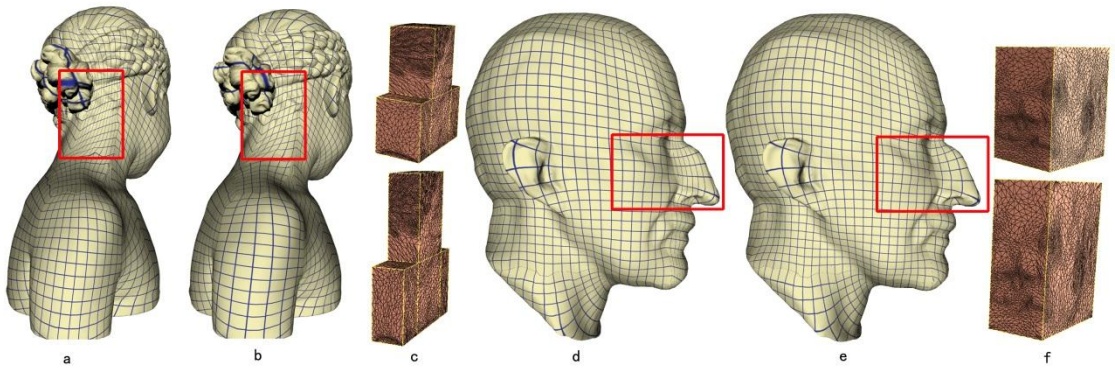

Figure 3.12. Polycube mapping of Bimba and Max-Planck.
(a,d) initial mapping, (b,e) optimized mapping. The texture mappings of grids show the reduction of angle distortions after the optimization. (c,f) initial polycube (in upper row) and optimized polycube (in lower row) domains.

Table 3-2. Comparisons of different polycube mapping methods.
***PC Constr., Opt. PC, Sing. Control, Common PC* indicate whether polycube construction can be automatic, whether polycube shape is optimal, whether polycube complexity can be controlled by the given restriction on singularity number, and whether it can be used to construct a canonical domain for multiple objects, respectively.**

Methods	PC Constr.	Opt. PC	Sing. Control	Common PC
Tarini[33]	Manual	No	Manual	No
Wang[34]	Manual	No	Manual	No
Wang[35]	Manual	No	Manual	No
Lin[24]	Auto.	No	No	No
He[18]	Auto.	No	Yes	No
Ours	Auto.	Yes	Yes	Yes

Figure 3.13 shows a common polycube parameterization for multiple objects. We parameterize the horse, cow, and goat onto an optimized common polycube domain. (a-c) visualize the geometry represented on the polycube parameterization (using the connectivity of the polycube), then we can easily interpolate them and generate a ``mixed creature". (d) shows an interpolated shape with $20\% - horse, 50\% - goat, and\ 30\% - cow$. Features of horse, goat, and cow can be seen on the final interpolated shape.

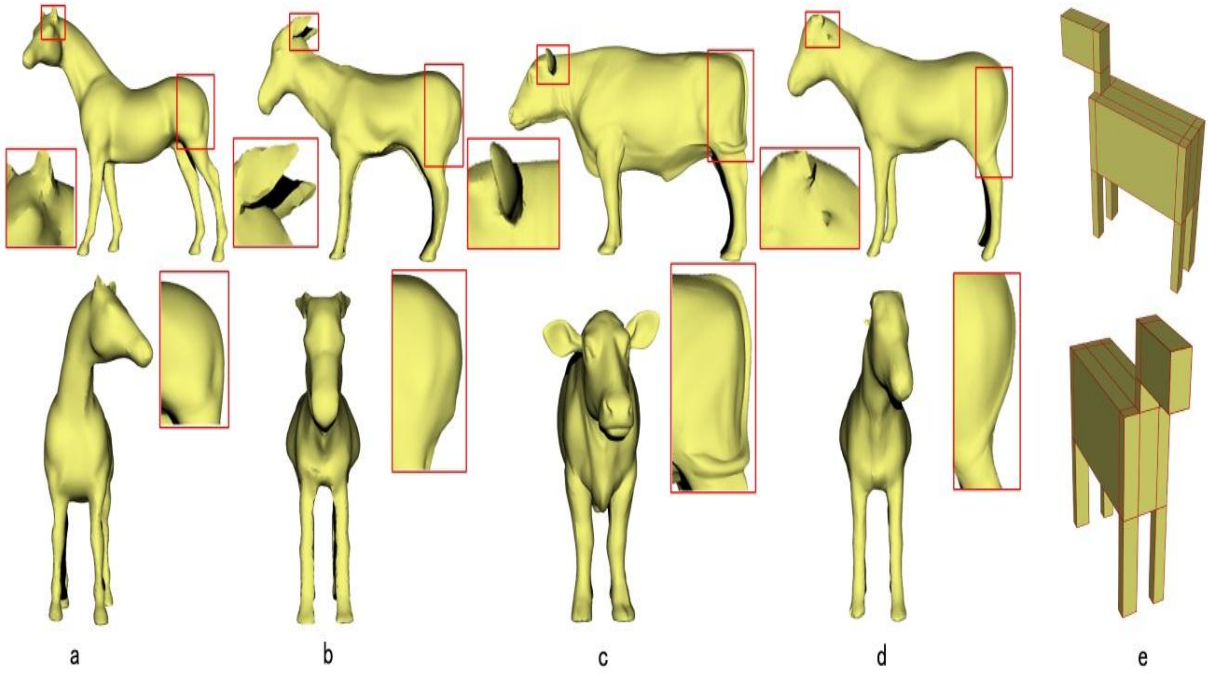

Figure 3.13. Integration of multiple objects over a common polycube domain. The horse (a), goat (b), and cow (c) are blended in this polycube domain. Features from the original models can still be seen in the interpolated shape (e.g. the mouth and neck of the horse, ears of the goat, and the tail of the cow).

The quality of polycube mapping can be measured by area distortion ϵ_{area} and angle distortion ϵ_{angle} [**32**].

$$\epsilon_{area}(T) = \frac{area(\Delta_{M'}(T))}{area(\Delta_M(T))} + \frac{area(\Delta_M(T))}{area(\Delta_{M'}(T))}$$

$$\epsilon_{angle} = \frac{\cot\alpha\,|a^2| + \cot\beta\,|b^2| + \cot\gamma\,|c^2|}{2area(\Delta_M(T))}$$

The closer the values of ϵ_{area} and ϵ_{angle} is to 1, the better the quality of polycube mapping we get. The statistics and performance of our test cases are reported in Table 3-3.

Intuitively, the more complicated the polycube domain is used, the more freedom we have to optimize its shape. And generally when the polycube is closer to the original model, we can get a less distorted/stretched polycube mapping. Figure 3.14 illustrates an example on the Beethoven model. When only one cube is used as the parameterization domain, the distortion is larger (a,b), compared with the mapping constructed on a more complicated

polycube domain (c,d). On the other hand, a more complicated polycube domain indicates more corner points (singularities) [**4**] and potentially more-distorted parameterization across sub-region boundaries.

Table 3-3. Runtime table
#Δ (number of triangles); #C number of corner points, ϵ^0_{angle} and ϵ^0_{area} are angle and area distortions before optimization; ϵ_{angle} and ϵ_{area} are distortions after optimization; $T1$ and $T2$ is the execution time for domain optimization and mapping optimization (in seconds).

Models	#Δ	#C	ϵ^0_{angle}	ϵ^0_{area}	ϵ_{angle}	ϵ_{area}	$T1$	$T2$
Isis	5K	8	1.261	1.429	1.134	1.385	0.52	112
Beethoven	21K	20	1.387	1.563	1.215	1.236	7.74	504
Max-Planck	10K	8	1.104	1.477	1.060	1.395	1.36	33
Bimba	30K	20	1.292	1.243	1.283	1.209	10.62	744
horse	16K	60	1.352	1.302	1.258	1.229	11.72	1842
cow	39K	60	1.198	1.210	1.191	1.161	21.21	2898
goat	21K	60	1.359	1.304	1.241	1.190	10.83	2032

We also adjust the weighting factor α in Equation 3.8 to see different mapping results. Table 3-4 shows the different angle and area distortion under different settings. $\alpha = 1.0$ was used when we perform our other experiments.

Figure 3.15 illustrates this mapping result. When the area term is emphasized, a more uniform but less conformal mapping is obtained (a,b); when α is small, the angle distortion is reduced (c,d).

Table 3-4. Testing different weighting on the area-stretching term. ϵ_{angle} and ϵ_{area} are the corresponding angle and area distortion. (α in Equation 3.8)

α	**0.1**	**0.5**	**1.0**	**1.5**
ϵ_{angle}	1.219	1.235	1.253	1.264
ϵ_{area}	1.380	1.316	1.292	1.281

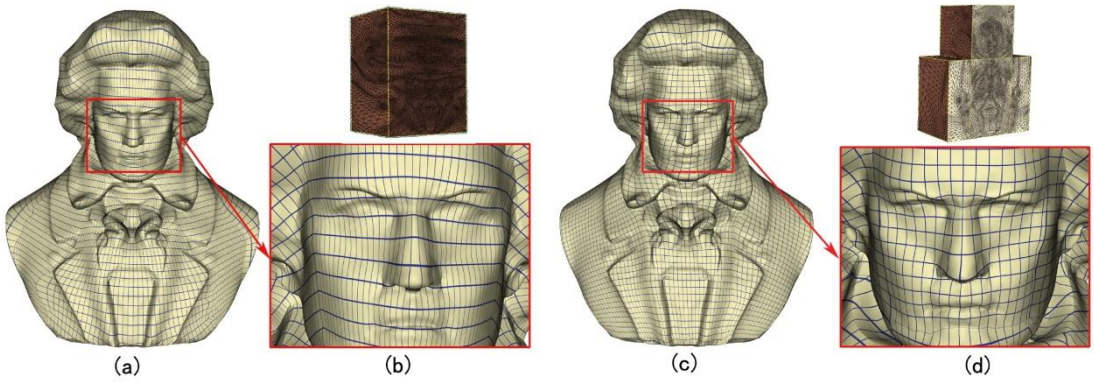

Figure 3.14. Different initial corner budgets.
With increase of the initial budget (from 8 to 20), the mapping quality is improved (from a,b to c,d).

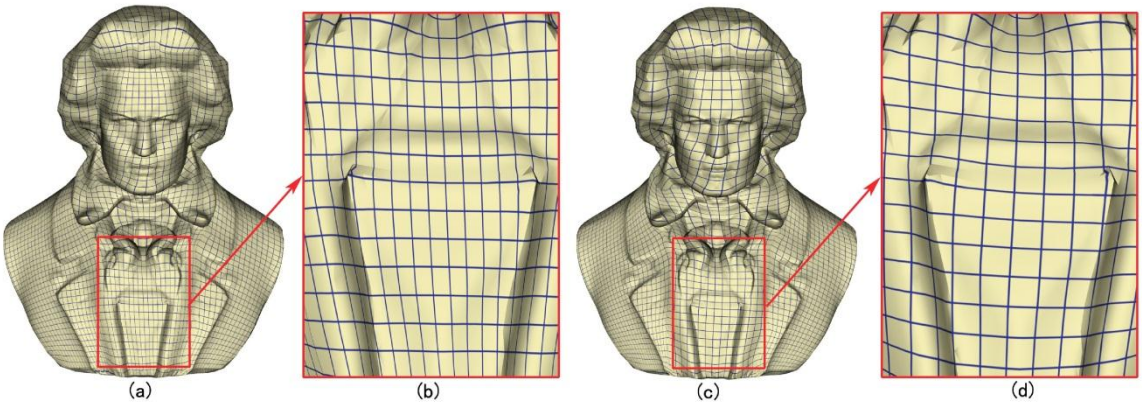

Figure 3.15. Different weighting factors.
(a,b) Area-stretching term $\alpha = 1000$, (c,d) $\alpha = 0.01$.

Chapter 4 Conclusion

4.1 Work Completed

Surface parameterization is a broad field and parameterization over regular domains, like polycube, attracts lots of attention of the researchers in recent years. Since Tarini et al. first introduced the concept of polycube mapping [**1**], much work has been done to improve polycube mapping. People were trying intrinsic approaches since the work of [**4**] rather than extrinsic approaches. Besides automation, researchers are trying to control the singularities and maintain a low-distortion parameterization independently. Unfortunately, the fact is we are going to have lower distortion mapping if the polycube domain approximates the input model more accurately, which means more corner points, or singularity points, are needed.

Obviously, a balance has to be achieved. This thesis proposes a question: what is the optimal polycube mapping given a singularity budget? Then it brings forward a solution. In this thesis, we consider the distortion and the singularity simultaneously. First of all, an initial polycube domain is constructed automatically for the input model, whose corner point number is constrained by the budget of singularities. After that, the polycube domain is optimized to have a polycube mapping with optimal distortion. Then the best corner point candidates are found through an optimization searching for minimized mapping distortion via local parameterization. The polycube domain optimization and mapping optimization run in an iterative way until the parameterization with no lower distortion can be obtained. During the optimization process, a fast approach to re-compute the parameterization and an efficient optimization solver is required. Following the strategies of [**53**], we utilize CHOLMOD [**52**] and create a fast polycube mapping updating scheme. Therefore, we do not have to recompute polycube mapping from scratch, which costs $O(n^3)$ each time. Moreover, we also

employ an efficient derivative-free solver to achieve the optimal polycube mapping since there is no close form for the objective function and it is difficult to obtain the derivatives of the objective function. We also apply this approach to common polycube domain for multiple objects, where the total distortion of all the polycube mappings among the objects is optimized. Experiments and demos have demonstrated the effectiveness of our approach.

4.2 Ongoing Work

Although this thesis proposes the new concepts of optimal polycube mapping and presents an effective solution, there exist some parts which can be improved in the future. For example, we adopt a simple polycube construction technique based on the octree and projection. It would not successful to extract the sub-patch information for complex models. What's more, allowing the alignment of feature points in the polycube mapping can benefit many graphics applications such as morphing and registration. However, this is challenging and has not been well discussed/solved in existing polycube mapping literature.

Within our current framework, on a sub-patch, directly enforcing the harmonic mapping to map an interior feature point to a specific position on the polycube domain may cause local flip-over around the feature point. One possible approach is to simply add feature alignment as a soft constraint in the mapping optimization step, such that feature matching errors are penalized like the angle-distortion and area-distortion terms. To enforce a hard constraint on feature matching, additional domain partitioning to make the features on the sub-patch boundary can be another solution. There are many problems here to explore.

Our research in spherical mapping is still in progress. We are developing a multi-resolution optimization approach to minimize energies defined on spherical meshes. The multi-resolution model representation has been implemented and we are now searching for an efficient way to insert and update new vertices into the existing optimized coarse mesh.

References

[1] Marco Tarini, Kai Hormann, Paolo Cignoni, and Claudio Montani, "PolyCube-Maps," *ACM Transactions on Graphics*, pp. 853--860, 2004.

[2] Hongwei Li, Kui-Yip Lo, Man-Kang Leung, and Chi-Wing Fu, "Dual Poisson-Disk Tiling: An Efficient Method for Distributing Features on Arbitrary Surfaces," *IEEE Transactions on Vision and Computer Graphics*, vol. 14, no. 5, pp. 982--998, 2008.

[3] Zhengwen Fan, Xiaogang Jin, Jieqing Feng, and Hanqiu Sun, "Mesh morphing using polycube-based cross-parameterization: Animating Geometrical Models," *Comput. Animat. Virtual Worlds*, vol. 16, no. 3-4, pp. 499--508, 2005.

[4] Hongyu Wang, Ying He, Xin Li, Xianfeng Gu, and Hong Qin, "Polycube splines," in *ACM SPM*, 2007, pp. 241--251.

[5] Hongyu Wang, Miao Jin, Ying He, Xianfeng Gu, and Hong Qin, "User-controllable polycube map for manifold spline construction," in *ACM SPM*, 2008, pp. 397--404.

[6] Xin Li et al., "Harmonic volumetric mapping for solid modeling applications," in *ACM SPM*, 2007, pp. 109--120.

[7] Xin Li, Huanhuan Xu, Shenghua Wan, Zhao Yin, and Wuyi Yu, "Feature-aligned Harmonic Volumetric Mapping using MFS," *Computers & Graphics*, vol. 34, no. 3, pp. 242-251, 2010.

[8] Xiaohu Guo, Xin Li, Yunfan Bao, Xianfeng Gu, and Hong Qin, "Meshless Thin-Shell Simulation Based on Global Conformal Parameterization," *IEEE Transactions on Vision and Computer Graphics*, vol. 12, no. 3, pp. 375-385, 2006.

[9] M. S. Floater and K. Hormann, "Surface Parameterization: a Tutorial and Survey," *Advances in Multiresolution for Geometric Modelling*, pp. 157-186, 2005.

[10] Alla Sheffer, Emil Praun, and Kenneth Rose, "Mesh parameterization methods and their applications," *Found. Trends. Comput. Graph. Vis.*, vol. 2, no. 2, pp. 105-171, 2006.

[11] Kai Hormann, Bruno Levy, and Alla Sheffer, "Mesh Parameterization: Theory and Practice," *ACM Siggraph 2007 Course*, vol. 11, pp. 1-87, 2007.

[12] U. Pinkall and K. Polthier, "Computing Discrete Minimal surfaces and their conjugates," *Experimental Mathematics*, vol. 2, pp. 15-36, 1993.

[13] Matthias Eck et al., "Multiresolution analysis of arbitrary meshes," in *ACM SIGRPAPH*, 1995, pp. 173-182.

[14] M. Desbrun, M. Meyer, and P. Alliez, "Intrinsic Parameterizations of Surface Meshes," *Computer Graphics Forum*, vol. 21, no. 3, pp. 209-218, 2002.

[15] M. S. Floater, "Mean value coordinates," *Computer Aided Geometric Design*, vol. 20, pp. 19-27, 2003.

[16] J. Lin, X. Jin, Z. Fan, and C. C. L. Wang, "Automatic PolyCube-Maps," in *Gemetric Modeling and Processing*, 2008, pp. 3-16.

[17] Y. He, H. Wang, C.-W. Fu, and H. Qin, "A divide-and-conquer approach for automatic polycube map construction," *Computers & Graphics*, vol. 33, no. 3, pp. 369-380, 2009.

[18] Shuchu Han, Jiazhi Xia, and Ying He, "Hexahedral shell mesh construction via volumetric polycube map," in *ACM SPM*, 2010, pp. 127--136.

[19] Jiazhil Xia, Ismael Garcia, Ying He, Shi-Qing Xin, and Gustavo Patow, "Editable polycube map for GPU-based subdivision surfaces," in *Symposium on Interactive 3D Graphics and Games*, 2011, pp. 151--158.

[20] H. Zhang, A. R. Conn, and K. Scheinberg, "A derivative-free algorithm for the least-squares minimization," *SIAM Journal on Optimization*, vol. 20, no. 6, pp. 3555-3576, 2010.

[21] W. Tutte, "Convex representation of graphs," in *London Math. Soc*, vol. 10, 1960.

[22] Mario Botsch, Leif Kobbelt, Mark Pauly, Pierre Alliez, and Bruno Levy, *Polygon Mesh Processing*.: AK Peters, 2010.

[23] M. S. Floater, "Parametrization and smooth approximation of surface triangulations," *Computer Aided Geometric Design*, vol. 14, no. 3, pp. 231-250, 1997.

[24] Bruno Levy, Sylvain Petitjean, Nicolas Ray, and Jerome Maillot, "Least squares conformal maps for automatic texture atlas generation," *ACM Transactions on Graphics*, vol. 21, no. 3, pp. 362-371, 2002.

[25] A. Sheffer and E. De Sturler, "Surface Parameterization for Meshing by Triangulation Flattening," in *9th International Meshing Roundtable*, 2000, pp. 161-172.

[26] A. Sheffer, B. Lévy, M. Mogilnitsky, and A. Bogomyakov, "ABF++: fast and robust angle based flattening," *ACM Transactions on Graphics*, vol. 24, pp. 311-330, 2005.

[27] Rhaleb Zayer, Bruno Levy, and Hans-Peter Seidel, "Linear angle based parameterization," in *Eurographics symposium on Geometry processing*, 2007, pp. 135-141.

[28] K. Hormann and G. Greiner, "MIPS: An Efficient Global Parametrization Method," *Curve and Surface Design*, pp. 153-162, 2000.

[29] Liliya Kharevych, Boris Springborn, and Peter Schröder, "Discrete conformal mappings via circle patterns," *ACM Transactions on Graphics*, vol. 25, pp. 412-438, 2006.

[30] Pedro V. Sander, John Snyder, Steven J. Gortler, and Hugues Hoppe, "Texture mapping progressive meshes," in *ACM SIGGRAPH*, 2001, pp. 409-416.

[31] G. Zigelman, R. Kimmel, and N. Kiryati, "Texture mapping using surface flattening via multidimensional scaling," *IEEE Transactions on Visualization and Computer Graphics*, vol. 8,

no. 2, pp. 198-207, 2002.

[32] P. Degener, J. Meseth, and R. Klein, "An Adaptable Surface Parameterization Method," in *International Meshing Roundtable*, 2003, pp. 201-213.

[33] Xianfeng Gu and Shing-tung Yau, "Computing conformal structures of surfaces," *Communications in Information and Systems*, vol. 2, pp. 121-146, 2002.

[34] S. Haker et al., "Conformal surface parameterization for texture mapping," *IEEE Transactions on Visualization and Computer Graphics*, vol. 6, no. 2, pp. 181-189, 2000.

[35] Xianfeng Gu and Shing-Tung Yau, "Global conformal surface parameterization," in *Eurographics/ACM SIGGRAPH symposium on Geometry processing*, 2003, pp. 127-137.

[36] M. Alexa, "Merging polyhedral shapes with scattered features," *The Visual Computer*, vol. 16, no. 1, pp. 26-37, 2000.

[37] L. P. Kobbelt, J. Vorsatz, U. Labsik, and H.-P. Seidel, "A shrink-wrapping approach," *Computer Graphics Forum*, vol. 18, no. 3, pp. 119-130, 1999.

[38] M. Isenburg, S. Gumhold, and C. Gotsman, "Connectivity shapes," in *IEEE Visualization*, 2001, pp. 135-142.

[39] Shadi Saba, Irad Yavneh, Craig Gotsman, and Alla Sheffer, "Practical Spherical Embedding of Manifold Triangle Meshes," in *International Conference on Shape Modeling and Applications*, 2005, pp. 258-267.

[40] Craig Gotsman, Xianfeng Gu, and Alla Sheffer, "Fundamentals of spherical parameterization for 3D meshes," *ACM Transactions on Graphics*, vol. 22, no. 3, pp. 358-363, 2003.

[41] Avner Shapiro and Ayellet Tal, "Polyhedron Realization for Shape Transformation," *The Visual Computer*, vol. 14, no. 8-9, pp. 429-444, 1998.

[42] Emil Praun and Hugues Hoppe, "Spherical parametrization and remeshing," *ACM Transactions on Graphics*, vol. 22, no. 3, pp. 340-349, 2003.

[43] A. Sheffer, C. Gotsman, and N. Dyn, "Robust spherical parametrization of triangular meshes," *Computing*, vol. 72, no. 1-2, 2004.

[44] E Praun, W. Sweldens, and P. Schroder, "Consistent mesh parameterizations," in *ACM SIGGRAPH*, 2001, pp. 179-184.

[45] Jonathan Barzilai and Jonathan M. Borwein, "Two-Point Step Size Gradient Methods," *IMA Journal of Numerical Analysis*, vol. 8, no. 1, pp. 141-148, 1988.

[46] Ernesto G. Birgin, Mario Mart Jos, and Marcos Raydan, "Nonmonotone spectral projected gradient methods on convex sets," *SIAM Journal on Optimization*, pp. 1196-1211, 2000.

[47] Yu-Hong Dai, William W. Hager, Klaus Schittkowski, and Hongchao Zhang, "The cyclic Barzilai-Borwein method for unconstrained optimization," *SIAM Journal on Optimization*, vol. 17, no. 2, pp. 526-557, 2006.

[48] Yu-Hong Dai and Hongchao Zhang, "Adaptive Two-Point Stepsize Gradient Algorithm," *Numerical Algorithms*, vol. 27, no. 4, pp. 377-385, 2001.

[49] William W. Hager and Hongchao Zhang, "A New Active Set Algorithm for Box Constrained Optimization," *SIAM Journal on Optimization*, vol. 17, no. 2, pp. 526-557, 2006.

[50] Shen Dong, Peer-Timo Bremer, Michael Garland, Valerio Pascucci, and John C. Hart, "Spectral surface quadrangulation," *ACM Transactions on Graphics*, vol. 25, no. 3, pp. 1057-1066, 2006.

[51] V. Kraevoy and Sheffer A., "Cross-parameterization and compatible remeshing of 3D models," *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 861-869, 2004.

[52] Timothy A. Davis and William W. Hager, "Dynamic Supernodes in Sparse Cholesky Update/Downdate and Triangular Solves," *ACM Transactons on Math. Softw.*, vol. 35, no. 4, pp. 27:1--27:23, 2009.

[53] Kai Xu, Hao Zhang, Daniel Cohen-Or, and Yueshan Xiong, "Dynamic harmonic fields for surface processing," *Computers & Graphics*, vol. 33, no. 3, pp. 391-398, 2009.

[54] M. J. D. Powell, "Least Frobenius norm updating of quadratic models that satisfy interpolation conditions," *Math. Program.*, vol. 100, no. 1, pp. 183--215, 2004.

[55] Jorge J. More and Stefan M. Wild, "Benchmarking derivative-free optimization algorithms," *SIAM Journal on Optimization*, vol. 20, no. 1, pp. 172-191, 2009.

[56] Jorge More, "The Levenberg-Marquardt algorithm: Implementation and theory," in *Numerical Analysis in Lecture Notes in Mathematics*., 1978, vol. 630, pp. 105-116.

[57] Charles Audet and J. E. Dennis Jr., "A Pattern Search Filter Method for Nonlinear Programming without Derivatives," *SIAM Journal on Optimization*, vol. 14, no. 4, pp. 980-1010, 2004.

[58] Xin Li, Xianfeng Gu, and Hong Qin, "Surface Mapping Using Consistent Pants Decomposition," *IEEE Transactions on Vision and Computer Graphics*, vol. 15, no. 4, pp. 558-571, 2009.

Vita

Shenghua Wan was born in the city of Nanchang of Jiangxi Province in People's Republic of China on April, 1987. He obtained the bachelor's degree in computer science from Harbin Institute of Technology, Harbin, Heilongjiang Province, People's Republic of China in July, 2009. One month later, He joined the doctoral program in the department of Electrical and Computer Engineering in Louisiana State University, Baton Rouge, United States of America. This master's thesis is finished in the middle of pursuing the doctoral degree for the commencement of fall semester of 2011 held in December, 2011.